z/OS Communications Server

IBM

IP Configuration Reference

Version 2 Release 1

SC27-3651-03

Note:
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 1493.

This edition applies to Version 2 Release 1 of z/OS (5650-ZOS), and to subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send us comments electronically by using one of the following methods:

Internet email:
comsvrcf@us.ibm.com

World Wide Web:
http://www.ibm.com/systems/z/os/zos/webqs.html

If you would like a reply, be sure to include your name, address, and telephone number. Make sure to include the following information in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this document

This document contains reference material such as statement syntax, options, keywords, and descriptions for z/OS® Communications Server. It also provides detailed information for the statements used to configure address spaces, servers, and applications. For detailed information about configuration-related tasks, see z/OS Communications Server: IP Configuration Guide.

Use this document to perform the following tasks:
- Configure z/OS Communications Server
- Customize and administer z/OS Communications Server

The information in this document includes descriptions of support for both IPv4 and IPv6 networking protocols. Unless explicitly noted, descriptions of IP protocol support concern IPv4. IPv6 support is qualified within the text.

This document refers to Communications Server data sets by their default SMP/E distribution library name. Your installation might, however, have different names for these data sets where allowed by SMP/E, your installation personnel, or administration staff. For instance, this document refers to samples in SEZAINST library as simply in SEZAINST. Your installation might choose a data set name of SYS1.SEZAINST, CS390.SEZAINST or other high level qualifiers for the data set name.

Who should read this document

This document is intended for programmers and system administrators who are familiar with TCP/IP, MVS™, z/OS, UNIX, and the Time Sharing Option Extensions (TSO/E).

How this document is organized

This document contains the following information:
- TCP/IP system information, including TCP/IP concepts and overview information about the TCP/IP system.
- Server application information, including descriptions of server applications, including cataloged procedures, and configuration statements.
- Appendixes provide additional details for the base and application information.
- “Notices” on page 1493 contains notices and trademarks used in this information.
- “Bibliography” on page 1503 contains descriptions of the information in the z/OS Communications Server library.

How to use this document

To use this document, you should be familiar with z/OS TCP/IP Services and the TCP/IP suite of protocols.

Determining whether a publication is current

As needed, IBM updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager® softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. The following information describes how to determine if you are looking at the most current copy of a publication:

- At the end of a publication's order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.
- If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
- To compare softcopy publications, you can check the last 2 characters of the publication's file name (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk (*) that indicates whether a publication is new or changed.

How to contact IBM service

For immediate assistance, visit this website: http://www.software.ibm.com/network/commserver/support/

Most problems can be resolved at this website, where you can submit questions and problem reports electronically, and access a variety of diagnosis information.

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-IBM-SERV). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

If you would like to provide feedback on this publication, see “Communicating your comments to IBM” on page 1539.

Conventions and terminology that are used in this document

Commands in this book that can be used in both TSO and z/OS UNIX environments use the following conventions:

- When describing how to use the command in a TSO environment, the command is presented in uppercase (for example, NETSTAT).
- When describing how to use the command in a z/OS UNIX environment, the command is presented in bold lowercase (for example, **netstat**).
- When referring to the command in a general way in text, the command is presented with an initial capital letter (for example, Netstat).

All the exit routines described in this document are *installation-wide exit routines*. The installation-wide exit routines also called installation-wide exits, exit routines, and exits throughout this document.

The TPF logon manager, although included with VTAM®, is an application program; therefore, the logon manager is documented separately from VTAM.

Samples used in this book might not be updated for each release. Evaluate a sample carefully before applying it to your system.

Note: In this information, you might see the following Shared Memory Communications over Remote Direct Memory Access (SMC-R) terminology:
- RDMA network interface card (RNIC), which is used to refer to the IBM® 10GbE RoCE Express® feature.
- Shared RoCE environment, which means that the 10GbE RoCE Express feature operates on an IBM z13™ (z13) or later system, and that the feature can be used concurrently, or shared, by multiple operating system instances. The RoCE Express feature is considered to operate in a shared RoCE environment even if you use it with a single operating system instance.

For definitions of the terms and abbreviations that are used in this document, you can view the latest IBM terminology at the IBM Terminology website.

Clarification of notes

Information traditionally qualified as Notes is further qualified as follows:

Note	Supplemental detail
Tip	Offers shortcuts or alternative ways of performing an action; a hint
Guideline	Customary way to perform a procedure
Rule	Something you must do; limitations on your actions
Restriction	Indicates certain conditions are not supported; limitations on a product or facility
Requirement	Dependencies, prerequisites
Result	Indicates the outcome

How to read a syntax diagram

This syntax information applies to all commands and statements that do not have their own syntax described elsewhere.

The syntax diagram shows you how to specify a command so that the operating system can correctly interpret what you type. Read the syntax diagram from left to right and from top to bottom, following the horizontal line (the main path).

Symbols and punctuation

The following symbols are used in syntax diagrams:

Symbol	**Description**
►►	Marks the beginning of the command syntax.
►	Indicates that the command syntax is continued.
|	Marks the beginning and end of a fragment or part of the command syntax.
►◄	Marks the end of the command syntax.

You must include all punctuation such as colons, semicolons, commas, quotation marks, and minus signs that are shown in the syntax diagram.

Commands

Commands that can be used in both TSO and z/OS UNIX environments use the following conventions in syntax diagrams:

- When describing how to use the command in a TSO environment, the command is presented in uppercase (for example, NETSTAT).
- When describing how to use the command in a z/OS UNIX environment, the command is presented in bold lowercase (for example, netstat).

Parameters

The following types of parameters are used in syntax diagrams.

Required
Required parameters are displayed on the main path.

Optional
Optional parameters are displayed below the main path.

Default
Default parameters are displayed above the main path.

Parameters are classified as keywords or variables. For the TSO and MVS console commands, the keywords are not case sensitive. You can code them in uppercase or lowercase. If the keyword appears in the syntax diagram in both uppercase and lowercase, the uppercase portion is the abbreviation for the keyword (for example, OPERand).

For the z/OS UNIX commands, the keywords must be entered in the case indicated in the syntax diagram.

Variables are italicized, appear in lowercase letters, and represent names or values you supply. For example, a data set is a variable.

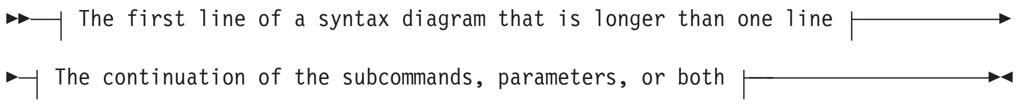
Syntax examples

In the following example, the PUt subcommand is a keyword. The required variable parameter is *local_file*, and the optional variable parameter is *foreign_file*. Replace the variable parameters with your own values.

```
►►──PUt──local_file──┬──────────────┬──────────────────────────────►◄
                     └─foreign_file─┘
```

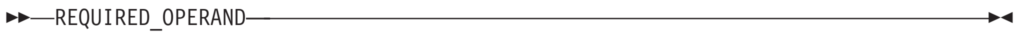
Longer than one line

If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead.


The first line of a syntax diagram that is longer than one line
The continuation of the subcommands, parameters, or both

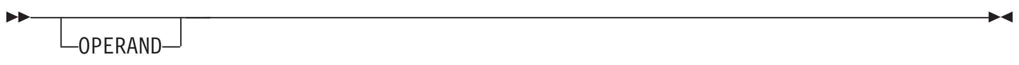
Required operands

Required operands and values appear on the main path line. You must code required operands and values.



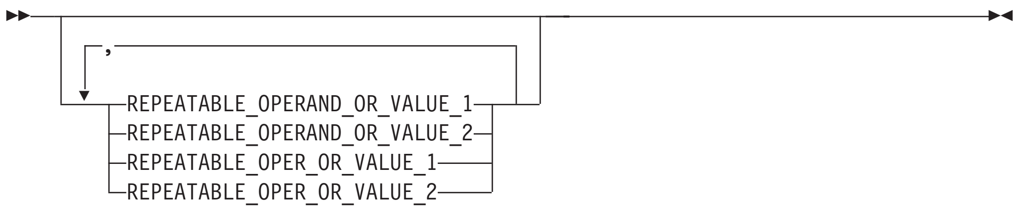
Optional values

Optional operands and values appear below the main path line. You do not have to code optional operands and values.



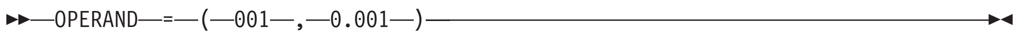
Selecting more than one operand

An arrow returning to the left above a group of operands or values means more than one can be selected, or a single one can be repeated.


,
REPEATABLE_OPERAND_OR_VALUE_1
REPEATABLE_OPERAND_OR_VALUE_2
REPEATABLE_OPER_OR_VALUE_1
REPEATABLE_OPER_OR_VALUE_2

Nonalphanumeric characters

If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code OPERAND=(001,0.001).



Blank spaces in syntax diagrams

If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code OPERAND=(001 FIXED).

```
►►──OPERAND──=──(──001── ──FIXED──)──────────────────────────────────────►◄
```

Default operands

Default operands and values appear above the main path line. TCP/IP uses the default if you omit the operand entirely.

```
     ┌─DEFAULT─┐
►►───┤         ├─────────────────────────────────────────────────────────►◄
     └─OPERAND─┘
```

Variables

A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

```
►►──variable─────────────────────────────────────────────────────────────►◄
```

Syntax fragments

Some diagrams contain syntax fragments, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

```
►►──┤ Syntax fragment ├──────────────────────────────────────────────────►◄
```

Syntax fragment:

```
├──1ST_OPERAND──,──2ND_OPERAND──,──3RD_OPERAND───────────────────────────┤
```

Prerequisite and related information

z/OS Communications Server function is described in the z/OS Communications Server library. Descriptions of those documents are listed in “Bibliography” on page 1503, in the back of this document.

Required information

Before using this product, you should be familiar with TCP/IP, VTAM, MVS, and UNIX System Services.

Softcopy information

Softcopy publications are available in the following collection.

Titles	Order Number	Description
IBM System z Redbooks Collection	SK3T-7876	The IBM Redbooks® publications selected for this CD series are taken from the IBM Redbooks inventory of over 800 books. All the Redbooks publications that are of interest to the System z® platform professional are identified by their authors and are included in this collection. The System z subject areas range from e-business application development and enablement to hardware, networking, Linux, solutions, security, parallel sysplex, and many others. For more information about the Redbooks publications, see http://www-03.ibm.com/systems/z/os/zos/zfavorites/.

Other documents

This information explains how z/OS references information in other documents.

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see z/OS Information Roadmap (SA23-2299). The Roadmap describes what level of documents are supplied with each release of z/OS Communications Server, and also describes each z/OS publication.

| To find the complete z/OS library, visit the z/OS library in IBM Knowledge Center
| (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

Relevant RFCs are listed in an appendix of the IP documents. Architectural specifications for the SNA protocol are listed in an appendix of the SNA documents.

The following table lists documents that might be helpful to readers.

Title	Number
DNS and BIND, Fifth Edition, O'Reilly Media, 2006	ISBN 13: 978-0596100575
Routing in the Internet, Second Edition, Christian Huitema (Prentice Hall 1999)	ISBN 13: 978-0130226471
sendmail, Fourth Edition, Bryan Costales, Claus Assmann, George Jansen, and Gregory Shapiro, O'Reilly Media, 2007	ISBN 13: 978-0596510299
SNA Formats	GA27-3136
TCP/IP Illustrated, Volume 1: The Protocols, W. Richard Stevens, Addison-Wesley Professional, 1994	ISBN 13: 978-0201633467
TCP/IP Illustrated, Volume 2: The Implementation, Gary R. Wright and W. Richard Stevens, Addison-Wesley Professional, 1995	ISBN 13: 978-0201633542
TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols, W. Richard Stevens, Addison-Wesley Professional, 1996	ISBN 13: 978-0201634952
TCP/IP Tutorial and Technical Overview	GG24-3376
Understanding LDAP	SG24-4986
z/OS Cryptographic Services System SSL Programming	*SC14-7495*
z/OS IBM Tivoli Directory Server Administration and Use for z/OS	*SC23-6788*

Title	Number
z/OS JES2 Initialization and Tuning Guide	*SA32-0991*
z/OS Problem Management	*SC23-6844*
z/OS MVS Diagnosis: Reference	*GA32-0904*
z/OS MVS Diagnosis: Tools and Service Aids	*GA32-0905*
z/OS MVS Using the Subsystem Interface	*SA38-0679*
z/OS V2R1 Program Directory	*GI11-9848*
z/OS UNIX System Services Command Reference	*SA23-2280*
z/OS UNIX System Services Planning	*GA32-0884*
z/OS UNIX System Services Programming: Assembler Callable Services Reference	*SA23-2281*
z/OS UNIX System Services User's Guide	*SA23-2279*
z/OS XL C/C++ Runtime Library Reference	*SC14-7314*
zEnterprise System and System z10 OSA-Express Customer's Guide and Reference	*SA22-7935*

Redbooks publications

The following Redbooks publications might help you as you implement z/OS Communications Server.

Title	Number
IBM z/OS V2R1 Communications Server TCP/IP Implementation, Volume 1: Base Functions, Connectivity, and Routing	SG24-8096
IBM z/OS V2R1 Communications Server TCP/IP Implementation, Volume 2: Standard Applications	SG24-8097
IBM z/OS V2R1 Communications Server TCP/IP Implementation, Volume 3: High Availability, Scalability, and Performance	SG24-8098
IBM z/OS V2R1 Communications Server TCP/IP Implementation, Volume 4: Security and Policy-Based Networking	SG24-8099
IBM Communication Controller Migration Guide	SG24-6298
IP Network Design Guide	SG24-2580
Managing OS/390 TCP/IP with SNMP	SG24-5866
Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender	SG24-5957
SecureWay Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements	SG24-5631
SNA and TCP/IP Integration	SG24-5291
TCP/IP in a Sysplex	SG24-5235
TCP/IP Tutorial and Technical Overview	GG24-3376
Threadsafe Considerations for CICS	SG24-6351

Where to find related information on the Internet

z/OS

This site provides information about z/OS Communications Server release availability, migration information, downloads, and links to information about z/OS technology

http://www.ibm.com/systems/z/os/zos/

z/OS Internet Library

Use this site to view and download z/OS Communications Server documentation

www.ibm.com/systems/z/os/zos/bkserv/

IBM Communications Server product

The primary home page for information about z/OS Communications Server

http://www.software.ibm.com/network/commserver/

IBM Communications Server product support

Use this site to submit and track problems and search the z/OS Communications Server knowledge base for Technotes, FAQs, white papers, and other z/OS Communications Server information

http://www.software.ibm.com/network/commserver/support/

IBM Communications Server performance information

This site contains links to the most recent Communications Server performance reports.

http://www.ibm.com/support/docview.wss?uid=swg27005524

IBM Systems Center publications

Use this site to view and order Redbooks publications, Redpapers™, and Technotes

http://www.redbooks.ibm.com/

IBM Systems Center flashes

Search the Technical Sales Library for Techdocs (including Flashes, presentations, Technotes, FAQs, white papers, Customer Support Plans, and Skills Transfer information)

http://www.ibm.com/support/techdocs/atsmastr.nsf

| **Tivoli NetView for z/OS**

| Use this site to view and download product documentation about Tivoli®
| NetView® for z/OS

| http://www.ibm.com/support/knowledgecenter/SSZJDU/welcome

RFCs

Search for and view Request for Comments documents in this section of the Internet Engineering Task Force website, with links to the RFC repository and the IETF Working Groups web page

http://www.ietf.org/rfc.html

Internet drafts

View Internet-Drafts, which are working documents of the Internet Engineering Task Force (IETF) and other groups, in this section of the Internet Engineering Task Force website

http://www.ietf.org/ID.html

Information about web addresses can also be found in information APAR II11334.

Note: Any pointers in this publication to websites are provided for convenience only and do not serve as an endorsement of these websites.

DNS websites

For more information about DNS, see the following USENET news groups and mailing addresses:

USENET news groups
comp.protocols.dns.bind

BIND mailing lists
https://lists.isc.org/mailman/listinfo

BIND Users
- Subscribe by sending mail to bind-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind-users@isc.org.

BIND 9 Users (This list might not be maintained indefinitely.)
- Subscribe by sending mail to bind9-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind9-users@isc.org.

The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS systems programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:
- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
- Make it easier for new people to learn z/OS

To access the z/OS Basic Skills Information Center, open your web browser to the following website, which is available to all users (no login required): http://www-01.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zbasics/homepage.html

Summary of changes

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Changes made in z/OS Version 2 Release 1, as updated February 2015

This document contains information previously presented in z/OS Communications Server: IP Configuration Reference, SC27-3651-02, which supported z/OS Version 2 Release 1.

Changed information

- Shared Memory Communications over RDMA adapter (RoCE) virtualization, see the following topics:
  - "GLOBALCONFIG statement" on page 117
  - "INTERFACE - IPAQENET OSA-Express QDIO interfaces statement" on page 145
  - "INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement" on page 161

Changes made in z/OS Version 2 Release 1, as updated September 2014

This document contains information previously presented in z/OS Communications Server: IP Configuration Reference, SC27-3651-01, which supported z/OS Version 2 Release 1.

Changes made in z/OS Version 2 Release 1, as updated December 2013

This document contains information previously presented in z/OS Communications Server: IP Configuration Reference, SC27-3651-00, which supported z/OS Version 2 Release 1.

Changed information

- AT-TLS enablement for DCAS, see Chapter 17, "EXPRESS LOGON using DCAS," on page 685.
- TLS security enhancements for policy agent, see the following topics:
  - "ServerConnection statement" on page 988
  - "ServicesConnection statement" on page 993
- Network security enhancements for SNMP, see the following topics:
  - "USM_USER entry" on page 1256
  - "SNMPD.CONF sample" on page 1272
  - "OSNMP.CONF statement syntax" on page 1281
  - "OSNMP.CONF sample" on page 1284

Summary of changes for z/OS Version 2 Release 1

For specifics on the enhancements for z/OS Version 2, Release 1, see the following publications:

- z/OS Summary of Message and Interface Changes
- z/OS Introduction and Release Guide
- z/OS Planning for Installation
- z/OS Migration

Chapter 1. Configuration data sets and files

This topic contains information about the configuration data sets and files that are used by the TCP/IP servers and functions.

This information refers to Communications Server data sets by their default SMP/E distribution library name. However, your installation might have different names for these data sets where allowed by SMP/E, your installation personnel, or administration staff. For instance, this topic refers to samples in hlq.SEZAINST library as simply in SEZAINST. Your installation might choose a data set name of SYS1.SEZAINST, CS390.SEZAINST or other high level qualifiers for the data set name.

The following terms are used in Table 1 on page 2:

hlq (high-level qualifier)
High-level qualifiers permit you to associate an application's configuration data set with a particular job name or TSO user ID, or permit you to use a default configuration data set for the application. The possible high-level qualifiers are:

userid The TSO user ID which invoked the application

jobname
The application's batch JCL JOB name or the name of the application's started procedure

default hlq
TCP/IP is distributed with a default hlq of TCPIP. To override the default used by dynamic data set allocation, specify the DATASETPREFIX statement in the TCPIP.DATA configuration file. For most servers or functions, the data set whose high-level qualifier matches the DATASETPREFIX value is the last data set in the search order. The data set whose high-level qualifier matches the DATASETPREFIX value is not the last in the search order for TCPIP.DATA configuration information.

SEZAINST (member)
Indicates that the sample is a member of the SEZAINST data set. This hlq value is the high-level qualifier specified during TCP/IP installation.

For some configuration information, the search order depends on the type of application (z/OS UNIX or native MVS). For a description of these search orders, see search orders used in the z/OS UNIX environment and search orders used in the native MVS environment in z/OS Communications Server: IP Configuration Guide.

TCP/IP configuration data sets

Table 1 on page 2 lists the configuration MVS data sets and z/OS UNIX files used by the TCP/IP servers and functions. The table includes the name of the sample data set or file that is provided by Communications Server, and the way the data set or file is used.

Table 1. TCP/IP configuration data sets

Name (search order)	Copied from	Usage
ADNR.CONF The MVS data set or z/OS UNIX file specified on the CONFIG DD statement in the automated domain name registration started procedure	SEZAINST(ADNRCNF)	Contains automated domain name registration configuration statements.
CSSMTP.CONF 1. The MVS data set or z/OS UNIX file referenced by the CONFIG DD statement in the CSSMTP application started procedure 2. *jobname*.CSSMTP.CONF	SEZAINST(CSSMTPCF)	Contains CSSMTP application configuration statements.
Defense Manager daemon (DMD) configuration 1. The MVS data set or z/OS UNIX file specified by the DMD_FILE environment variable 2. /etc/security/dmd.conf	/usr/lpp/tcpip/samples/dmd.conf	Contains DMD configuration statements.
Digital certificate access server (DCAS) configuration 1. The MVS data set or z/OS UNIX file that the DCAS_CONFIG_FILE environment variable specified 2. /etc/dcas.conf 3. *tsouserid*.DCAS.CONF 4. TCPIP.DCAS.CONF	No sample provided.	Contains DCAS configuration statements.
/etc/hosts	No sample provided.	One of the possible local host files used for IPv4 name query. For information about creating /etc/hosts directory, see z/OS Communications Server: IP Configuration Guide
hlq.ETC.IPNODES	SEZAINST(EZBREIPN)	One of the local host files used for IPv6 name query, or IPv4 and IPv6 name query when COMMONSEARCH is specified in the resolver setup file.

Table 1. TCP/IP configuration data sets (continued)

Name (search order)	Copied from	Usage
/etc/mail/sendmail.cf	/usr/lpp/tcpip/samples/sendmail/cf/sample.cf	Provides configuration information for the sendmail daemon when being used as a message transfer agent (MTA). If /etc/mail/submit.cf does not exist, this data set also provides configuration information for the end-user sendmail application when being used as a mail user agent (MUA).
/etc/mail/submit.cf	/usr/lpp/tcpip/samples/sendmail/cf/submit.cf	Provides configuration information for the end-user sendmail application when being used as a mail user agent (MUA).
/etc/mail/zOS.cf	/usr/lpp/tcpip/samples/sendmail/cf/zOS.cf	Provides z/OS-specific information for the sendmail daemon when being used as a message transfer agent (MTA). Currently the file consists of Secure Sockets Layer (SSL) information only.
ETC.PROTO	usr/lpp/tcpip/samples/protocol	Used to map types of protocol to integer values to determine the availability of the specified protocol. Required by several z/OS Communications Server components. The search order depends on the type of application (z/OS UNIX or native MVS).
ETC.RPC	SEZAINST(ETCRPC)	Defines RPC applications to the Portmapper function.
ETC.SERVICES	usr/lpp/tcpip/samples/services	Establishes port numbers for servers using TCP and UDP. Required for z/OS UNIX SNMP and OMPROUTE (if the RIP protocol is used). The search order depends on the type of application (z/OS UNIX or native MVS).
/etc/syslog.conf	/usr/lpp/tcpip/samples/syslog.conf	Configuration file for the syslog daemon (syslogd).

Table 1. TCP/IP configuration data sets (continued)

Name (search order)	Copied from	Usage
FTP.DATA 1. -f command line parameter (FTP client only) 2. The MVS data set or z/OS UNIX file specified on the SYSFTPD DD statement in the FTP server started procedure 3. *userid/jobname*.FTP.DATA 4. /etc/ftp.data 5. SYS1.TCPPARMS(FTPDATA) 6. *hlq*.FTP.DATA	SEZAINST(FTCDATA) for the client and (FTPSDATA) for the server	Overrides default FTP client and server parameters for the FTP server. For more information about the *hlq*, *jobname*, or *userid* values, see Chapter 18, “File Transfer Protocol,” on page 697.
HOSTS.LOCAL	SEZAINST(HOSTS)	Input data set to MAKESITE for generation of HOSTS.ADDRINFO and HOSTS.SITEINFO.
IKE daemon configuration 1. The MVS data set or z/OS UNIX file specified by the IKED_FILE environment variable 2. /etc/security/iked.conf	/usr/lpp/tcpip/samples/iked.conf	Contains IKE configuration statements.
INETD.CONF The MVS data set or z/OS UNIX file specified on the EXEC DD statement in the INETD started procedure	/samples/inetd.conf	Provides configuration management statements of generic servers for the Internet Daemon (InetD). InetD handles rlogin, telnetd, rshd, rexec, and other applications. For more information about InetD, see z/OS UNIX System Services Planning.
LBADV.CONF The MVS data set or z/OS UNIX file specified on the CONFIG DD statement in the z/OS Load Balancing Advisor started procedure	SEZAINST(LBADVCNF)	Contains z/OS Load Balancing Advisor configuration statements.
LBAGENT.CONF The MVS data set or z/OS UNIX file specified on the CONFIG DD statement in the z/OS Load Balancing Agent started procedure.	SEZAINST(LBAGECNF)	Contains z/OS Load Balancing Agent configuration statements.
LPD.CONFIG	SEZAINST(LPDDATA)	Configures the Line Printer Daemon for the Remote Print Server.
LU62CFG	SEZAINST(LU62CFG)	Provides configuration parameters for the SNALINK LU6.2 interface.

Table 1. TCP/IP configuration data sets (continued)

Name (search order)	Copied from	Usage
MIBS.DATA 1. The name of a z/OS UNIX file or an MVS data set specified by the MIBS_DATA environment variable 2. /etc/mibs.data z/OS UNIX file	No sample provided	Defines textual names for MIB objects for the z/OS UNIX snmp command.
Network security services (NSS) server configuration 1. The name of a z/OS UNIX file or MVS data set specified by the NSSD_FILE environment variable. 2. /etc/security/nssd.conf	/usr/lpp/tcpip/samples/nssd.conf	Contains NSS server configuration statements.
NPSIDATE	SEZAINST(NPSIDATE)	• Operates the TCP/IP X.25 NCP Packet Switching Interface. • NCP and X.25 definition statements supplied as input to the NCP/EP Definition Facility (NDF) procedure. See *NCP X.25 Planning and Installation* for details.
NPSIGATE	SEZAINST(NPSIGATE)	• Supports GATE MCHs for X.25 NCP Packet Switching Interface. • NCP and X.25 definition statements supplied as input to the NCP/EP Definition Facility (NDF) procedure. See *Network Control Program X.25 Planning and Installation* for details.
OMPROUTE configuration 1. The MVS data set or z/OS UNIX file specified on the OMPCFG DD statement in the OMPROUTE started procedure. 2. The MVS data set or z/OS UNIX file specified by the OMPROUTE_FILE environment variable 3. /etc/omproute.conf 4. *hlq*.ETC.OMPROUTE.CONF	SEZAINST(EZAORCFG)	Contains OMPROUTE configuration statements.
OSNMP.CONF 1. /etc/osnmp.conf 2. /etc/snmpv2.conf	/usr/lpp/tcpip/samples/snmpv2.conf	Defines target host security parameters for the osnmp command.

Table 1. TCP/IP configuration data sets (continued)

Name (search order)	Copied from	Usage
OSNMPD.DATA 1. The MVS data set or z/OS UNIX file specified by the OSNMPD_DATA environment variable 2. /etc/osnmpd.data file system file 3. The MVS data set z/OS UNIX file specified on the OSNMPD DD statement in the agent started procedure 4. *jobname*.OSNMPD.DATA, where *jobname* is the name of the job used to start the SNMP agent 5. SYS1.TCPPARMS(OSNMPD) 6. *hlq*.OSNMPD.DATA, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used	/usr/lpp/tcpip/samples/osnmpd.data	Used by SNMP for setting values for selected MIB objects.
PAGENT.CONF 1. File or data set specified with -c startup option 2. File or data set specified with PAGENT_CONFIG_FILE environment variable 3. /etc/pagent.conf	/usr/lpp/tcpip/samples/pagent.conf	Defines Policy Agent configuration parameters and optionally defines QoS service policies (rules and actions).
PROFILE.TCPIP 1. The MVS data set specified on the PROFILE DD statement in the TCP/IP stack started procedure. 2. *jobname.nodename*.TCPIP 3. TCPIP.*nodename*.TCPIP 4. *jobname*.PROFILE.TCPIP 5. TCPIP.PROFILE.TCPIP	SEZAINST(SAMPPROF)	Provides TCP/IP initialization parameters and specifications for network interfaces and routing.

Table 1. TCP/IP configuration data sets (continued)

Name (search order)	Copied from	Usage
PW.SRC 1. The MVS data set or z/OS UNIX file specified by the PW_SRC environment variable 2. /etc/pw.src file system file 3. The MVS data set or z/OS UNIX file specified on SYSPWSRC DD statement in the started agent procedure 4. *jobname*.PW.SRC, where *jobname* is the name of the job used to start the SNMP agent 5. SYS1.TCPPARMS(PWSRC) 6. *hlq*.PW.SRC, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used	No sample provided	Defines a list of community names used when accessing objects on a destination SNMP agent.
Resolver Setup File	SEZAINST (RESSETUP)	Provides configuration statements for the resolver.
RSVPD.CONF 1. File or data set specified with -c startup option 2. File or data set specified with RSVPD_CONFIG_FILE environment variable 3. /etc/rsvpd.conf 4. *hlq*.RSVPD.CONF	/usr/lpp/tcpip/samples/rsvpd.conf	Defines RSVP Agent configuration parameters.
SMTPCONF The MVS data set referenced by CONFIG DD statement in the SMTP started procedure.	SEZAINST(SMTPCONF)	Provides configuration parameters for the Simple Mail Transfer Protocol (SMTP).
SMTPNOTE clist System CLIST data set	SEZAINST(SMTPNOTE)	Defines the *note* parameters for Simple Mail Transfer Protocol (SMTP) and the CSSMTP application.

Table 1. TCP/IP configuration data sets (continued)

Name (search order)	Copied from	Usage
SNMPD.BOOTS 1. The name of a z/OS UNIX file system file or an MVS data set specified by the SNMPD_BOOTS environment variable. 2. /etc/snmpd.boots	No sample provided	Defines the SNMP agent security and notification destinations. **Note:** If the SNMPD.BOOTS file is not provided, the SNMP agent creates the file. If multiple SNMPv3 agents are running on the same MVS image, use the environment variable to specify different SNMPD.BOOTS files for the different agents. For security reasons, ensure unique engine IDs are used for different SNMP agents.
SNMPD.CONF 1. The name of a z/OS UNIX file system file or an MVS data set specified by the SNMPD_CONF environment variable. 2. /etc/snmpd.conf **Note:** The first file found in the search order is used.	/usr/lpp/tcpip/samples/snmpd.conf	Defines the SNMP agent security and notification destinations. **Note:** If the SNMPD.CONF file is found, the PW.SRC file and the SNMPTRAP.DEST files are not used.
SNMPTRAP.DEST 1. The MVS data set or z/OS UNIX file specified by the SNMPTRAP_DEST environment variable 2. /etc/snmptrap.dest file system file 3. The MVS data set or z/OS UNIX file specified on SNMPTRAP DD statement in the agent started procedure 4. *jobname*.SNMPTRAP.DEST, where *jobname* is the name of the job used to start the SNMP agent 5. SYS1.TCPPARMS(SNMPTRAP) 6. *hlq*.SNMPTRAP.DEST, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used	No sample provided	Defines a list of managers to which the SNMP agent sends traps.

Table 1. TCP/IP configuration data sets (continued)

Name (search order)	Copied from	Usage
TCPIP.DATA	SEZAINST(TCPDATA)	Provides parameters for TCP/IP client programs. The search order depends on the type of application (z/OS UNIX or native MVS). See Chapter 5, "Resolver setup and TCPIP.DATA configuration statements," on page 355 for more information.
TNDBCSCN The MVS data set specified on the TNDBCSCN DD statement in the TN3270E Telnet server started procedure	SEZAINST(TNDBCSCN)	Provides configuration parameters for Telnet 3270 Transform support.
TRAPFWD.CONF 1. A z/OS UNIX system file or an MVS data set specified by the TRAPFWD_CONF environment variable 2. /etc/trapfwd.conf	No sample provided	Defines addresses to which the Trap Forwarder Daemon forwards traps. **Note:** If the environment variable is set and if the file specified by the environment variable is not found, the Trap Forwarder daemon terminates.
VTAMLST The VTAM definitions added to the ATCCONxx member of the MVS data set specified on the VTAMLST DD statement in the VTAM started procedure	SEZAINST(VTAMLST)	Defines VTAM applications and their characteristics. Entries required for TN3270E Telnet server, SNALINK LU0, SNALINK LU6.2, and X.25 NPSI Server.
X25CONF The MVS data set specified on the X25IPI DD statement in the X25PROC started procedure	SEZAINST(X25CONF)	Provides configuration parameters for the X.25 NCP Packet Switching Interface.
X25VSVC The VTAM switched major node definition, added as a member of the MVS data set specified on the VTAMLST DD statement in the VTAM started procedure	SEZAINST(X25VSVC)	Provides switched virtual circuit configuration for the X.25 NCP Packet Switching Interface.

Chapter 2. TCP/IP profile (PROFILE.TCPIP) and configuration statements

This topic contains the following information:

- "Summary of TCP/IP address space configuration statements"
- "PROFILE.TCPIP search order" on page 14
- "Statement syntax for configuration statements" on page 15
- Statements and descriptions

Configuring the stack for IPv6 is done in the BPXPRMxx member of SYS1.PARMLIB. For more information about configuring the stack to support IPv6, see z/OS Communications Server: IP Configuration Guide or z/OS Communications Server: IPv6 Network and Application Design Guide.

Summary of TCP/IP address space configuration statements

Table 2 contains a brief description of each configuration statement, along with the location of more information.

Table 2. Summary of TCP/IP address space configuration statements

Statement	Description	See
ARPAGE	Alters the number of minutes before an ARP table entry is deleted.	"ARPAGE statement" on page 16
ATMARPSV	Defines the ATMARP server to resolve ATMARP requests for a logical IP subnetwork (LIS).	"ATMARPSV statement" on page 17
ATMLIS	Describes the characteristics of an ATM logical IP subnet (LIS).	"ATMLIS statement" on page 19
ATMPVC	Describes a permanent virtual circuit to be used by an ATM link.	"ATMPVC statement" on page 22
AUTOLOG	Indicates which procedures should be automatically started when TCP/IP is started.	"AUTOLOG statement" on page 23
BEGINROUTES, ENDROUTES	Defines main routing table entries in standard Berkeley Software Distribution (BSD) format for static routes.	"BEGINROUTES statement" on page 28
BSDROUTINGPARMS	Defines network interface information. Used to provide interface-level characteristics to interfaces used for static routing or NCPROUTE.	"BSDROUTINGPARMS statement" on page 36
DEFADDRTABLE	Defines the policy table for IPv6 default address selection.	"DEFADDRTABLE statement" on page 41
DELETE	Removes an ATMARPSV, ATMLIS, ATMPVC, device, link, port, or portrange.	"DELETE statement" on page 43

Table 2. Summary of TCP/IP address space configuration statements (continued)

Statement	Description	See
DEVICE and LINK statements	Defines an IPv4 device. To configure your devices, add the appropriate DEVICE and LINK statements to the configuration data set. The LINK statements show how to define a network interface link associated with the device and are included with the DEVICE statement for that device type.	"Summary of DEVICE and LINK statements" on page 47
DEVICE and LINK	ATM devices	"DEVICE and LINK — ATM devices statement" on page 52
DEVICE and LINK	CLAW devices	"DEVICE and LINK — CLAW devices statement" on page 55
DEVICE and LINK	CTC devices	"DEVICE and LINK — CTC devices statement" on page 60
DEVICE and LINK	HYPERchannel A220 devices	"DEVICE and LINK — HYPERchannel A220 devices statement" on page 63
DEVICE and LINK	LAN Channel Station and OSA devices	"DEVICE and LINK — LAN Channel Station and OSA devices statement" on page 66
DEVICE and LINK	MPCIPA devices	"DEVICE and LINK — MPCIPA OSA-Express QDIO devices statement" on page 74
DEVICE and LINK	MPCIPA HiperSockets™ devices	"DEVICE and LINK — MPCIPA HiperSockets devices statement" on page 85
DEVICE and LINK	MPCOSA devices	"DEVICE and LINK — MPCOSA devices statement" on page 89
DEVICE and LINK	MPCPTP devices Used for: • EE • HPDT • Communication between stacks • XCF connections	"DEVICE and LINK — MPCPTP devices statement" on page 92
DEVICE and LINK	SNA LU0 links	"DEVICE and LINK — SNA LU0 links statement" on page 95
DEVICE and LINK	SNA LU 6.2 links	"DEVICE and LINK — SNA LU 6.2 links statement" on page 98
DEVICE and LINK	X.25 NPSI connections	"DEVICE and LINK - X.25 NPSI connections statement" on page 103
DEVICE and LINK	Virtual devices	"DEVICE and LINK — VIRTUAL devices statement" on page 101
DEVICE and LINK	3745/46 Channel DLC devices	"DEVICE and LINK — 3745/46 channel DLC devices statement" on page 105
GATEWAY	Defines main routing table entries for static routes.	"GATEWAY statement" on page 109
GLOBALCONFIG	Passes global configuration parameters to TCP/IP.	"GLOBALCONFIG statement" on page 117

Table 2. Summary of TCP/IP address space configuration statements (continued)

Statement	Description	See
HOME	Provides a list of home addresses and associated link names.	“HOME statement” on page 136
INCLUDE	Causes another data set that contains profile configuration statements to be included at this point.	“INCLUDE statement” on page 141
INTERFACE statements	Defines an IPv4 interface for OSA-Express QDIO Ethernet, HiperSockets, or static VIPA, or defines an IPv6 interface.	“Summary of INTERFACE statements” on page 141
INTERFACE	IPAQENET interfaces Specifies IPv4 OSA-Express QDIO interfaces.	“INTERFACE - IPAQENET OSA-Express QDIO interfaces statement” on page 145
INTERFACE	IPAQENET6 interfaces Specifies IPv6 OSA-Express QDIO interfaces.	“INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement” on page 161
INTERFACE	IPAQIDIO interfaces Configures IPv4 HiperSockets connectivity.	“INTERFACE — IPAQIDIO HiperSockets interfaces statement” on page 157
INTERFACE	IPAQIDIO6 interfaces Configures IPv6 HiperSockets connectivity.	“INTERFACE — IPAQIDIO6 HiperSockets interfaces statement” on page 177
INTERFACE	LOOPBACK6 interface Allows you to add additional IP addresses for LOOPBACK6 in the initial profile or in a data set used with the VARY TCPIP,,OBEYFILE command.	“INTERFACE — LOOPBACK6 interface statement” on page 182
INTERFACE	MPC Point-to-Point interfaces Updated Data Link Control supports IPv6 traffic.	“INTERFACE — MPCPTP6 interfaces statement” on page 183
INTERFACE	VIRTUAL interfaces Specifies IPv4 static virtual interfaces.	“INTERFACE — VIRTUAL interfaces statement” on page 160
INTERFACE	VIRTUAL6 interfaces Specifies IPv6 static virtual interfaces.	“INTERFACE — VIRTUAL6 interfaces statement” on page 188
IPCONFIG	Specifies IP configuration values.	“IPCONFIG statement” on page 190
IPCONFIG6	Specifies IPv6 configuration values.	“IPCONFIG6 statement” on page 206
IPSEC	Specifies policy for the IP Security function.	“IPSEC statement” on page 218
ITRACE	Controls tracing for configuration, the SNMP subagent, commands, and the autolog subtask.	“ITRACE statement” on page 227
NETACCESS, ENDNETACCESS	Configures network access.	“NETACCESS statement” on page 229
NETMONITOR	Activates or deactivates network management programming interfaces.	“NETMONITOR statement” on page 234

Table 2. Summary of TCP/IP address space configuration statements (continued)

Statement	Description	See
OSAENTA	Defines the conditions used to select Ethernet frames from an OSA as candidates for tracing and subsequent analysis.	"OSAENTA statement" on page 241
PKTTRACE	Defines the conditions used to select IP packets as candidates for tracing and subsequent analysis.	"PKTTRACE statement" on page 250
PORT	Reserves a port for one or more given job names or controls application access to unreserved ports.	"PORT statement" on page 257
PORTRANGE	Reserves a range of ports for one or more job names.	"PORTRANGE statement" on page 266
PRIMARYINTERFACE	Specifies which link is to be considered the primary interface.	"PRIMARYINTERFACE statement" on page 270
SACONFIG	Specifies parameters for the TCP/IP SNMP subagent.	"SACONFIG statement" on page 271
SMFCONFIG	Provides SMF logging for Telnet, FTP, IPSec, TCP API, and TCP stack activity.	"SMFCONFIG statement" on page 274
SMFPARMS	Provides SMF logging for Telnet and FTP client activity and TCP API activity.	"SMFPARMS statement" on page 281
SOMAXCONN	Specifies a maximum connection length for the connection request queues created by the socket call listen().	"SOMAXCONN statement" on page 282
SRCIP	Designates source IP addresses to be used for outbound TCP connections that are initiated by specified jobs or destined for specified IP addresses, networks, or subnets.	"SRCIP statement" on page 282
START	Starts the specified device or interface.	"START statement" on page 292
STOP	Stops the specified device or interface.	"STOP statement" on page 293
TCPCONFIG	Specifies TCP parameters.	"TCPCONFIG statement" on page 294
TRANSLATE	Indicates the relationship between an IP address and the network address.	"TRANSLATE statement" on page 301
UDPCONFIG	Specifies UDP parameters.	"UDPCONFIG statement" on page 304
VIPADYNAMIC, ENDVIPADYNAMIC	Specifies a block of definitions related to dynamic VIPAs. This includes VIPABACKUP, VIPADEFINE, VIPADELETE, VIPADISTRIBUTE, VIPARANGE, and VIPASMPARMS.	"VIPADYNAMIC statement summary" on page 306

PROFILE.TCPIP search order

The search order for accessing PROFILE.TCPIP information is as follows. The first file found in the search order is used.

1. //PROFILE DD statement
2. *jobname.nodename*.TCPIP

3. TCPIP.*nodename*.TCPIP
4. *jobname*.PROFILE.TCPIP
5. TCPIP.PROFILE.TCPIP

Statement syntax for configuration statements

Statement syntax is the same in both the configuration data set *hlq*.PROFILE.TCPIP and the VARY TCPIP,,OBEYFILE command data set.

- Entries in a configuration data set are free format; blanks, comments, and end-of-record are ignored.
- A configuration statement consists of a statement name followed by a required blank, and usually one or more positional arguments. Separate each argument by one or more blanks or end-of-record.
- A semicolon begins a comment. Comments act as blanks, separating words without affecting their meaning.
- An argument followed by a comment must have a blank before the semicolon.
- Statements can be split across multiple lines.
- Sequence numbers are not allowed.
- Lowercase letters are translated to uppercase before the statements are executed, except for those parameters that support mixed case entries. For example, the SNMP community name is case sensitive.
- An END statement terminates a number of statements, such as AUTOLOG. If the END statement is omitted, all subsequent tokens in the data set are interpreted as parameters for that configuration statement.
- If a syntax error is encountered in a list of parameters, such as a HOME list, the rest of the entries in the list are ignored.

  Tip: Because some statements skip the entry in error and continue to process the remaining entries, this does not apply to all statements.
- Profile statements do have some order restrictions. The basic order is any statement that references a name defined in another statement must follow that statement. For example, LINK statements must follow the DEVICE statement that defines the device referenced by the link. Statements referencing links (for example, GATEWAY, HOME, and TRANSLATE) must follow the referenced LINK statement.
- Static system symbols can be used in profile statements.
- For those profile statements where you can modify parameters by respecifying the statement, the only parameter values that are changed when the statement is respecified are those parameters explicitly coded on the respecified statement. The parameter values that are not explicitly coded on the statement are not changed to the default value of the parameter; they retain their last specified value. For example, if you specify: IPCONFIG NODATAGRAMFWD in an initial profile data set, and then specify: IPCONFIG IGNOREREDIRECT in a data set referenced by a VARY TCPIP,,OBEYFILE command, the NODATAGRAMFWD parameter remains in effect and is not changed to the default parameter value of DATAGRAMFWD NOFWDMULTIPATH.
- **Rules:** User-defined names on configuration statements must adhere to the following rules:
  - Each character must be a non-blank printable character.
  - The following characters are not allowed:
    - Comma (,)
    - Semicolon (;)

        - Equal (=)
        - Asterisk (*)
    – The following characters are considered printable characters:
        - ¢ < (+ | & ! $) ¬ \ - / % _ > ? ` : # @ ' " Ý \ { } ¬ ~ (also alphabetic and numeric characters)
    – The first character must be alphanumeric or either $ or @. If it is numeric, the name must not be a hexadecimal number.
- IPv4 IP addresses can be partially defined in a Profile statement where an IP address is expected. If a user enters 100, it is interpreted as 100.0.0.0, and 1.2 is interpreted as 1.2.0.0.
- All characters must be entered in code page IBM-1047.

Guideline: Use the VARY TCPIP,,SYNTAXCHECK command to verify that the configuration statements in the configuration data set are free of syntax errors before starting the TCP/IP stack or activating a new profile using the Vary TCPIP,,OBEYFILE command to activate the profile. To use the syntax checker before starting the stack, you must issue the VARY TCPIP,,SYNTAXCHECK command on a system that has already started TCP/IP. See VARY TCPIP,,SYNTAXCHECK in z/OS Communications Server: IP System Administrator's Commands for more information.

ARPAGE statement

Use the ARPAGE statement to change the number of minutes between creation or revalidation of an ARP table entry, and deletion of the entry. By default, TCP/IP deletes ARP table entries 20 minutes after creation or revalidation. An ARP table entry is revalidated when another ARP packet is received from the same host specifying the same hardware address. The ARPAGE statement only applies to LAN channel station (LCS) devices.

Syntax

```
            ┌─20──────┐
►►──ARPAGE──┼─────────┼──────────────────────────────────────────────►◄
            └─minutes─┘
```

Parameters

minutes
The number of minutes between creation or revalidation of an ARP table entry and deletion of the entry.

This number is an integer in the range 1 - 1 440 (24 hours). The default is 20 minutes.

Steps for modifying

To modify parameters for the ARPAGE statement, you must respecify the statement with the new parameters.

Statement dependency

Because ARP cache entries for MPCIPA and MPCOSA interfaces are not managed by the TCP/IP stack, they are not affected by the ARPAGE statement.

Examples

This example causes revalidation of ARP table entries every 10 minutes.

```
ARPAGE 10
```

Usage notes

- IPCONFIG ARPTO allows you to specify the number of *seconds* between creation or revalidation and deletion.
- The revalidation of ARP requests for asynchronous transfer mode (ATM) is controlled using the ATMLIS statement.

Related topics

See ARPTO in “IPCONFIG statement” on page 190.

ATMARPSV statement

Because support will be eliminated for the ATM device type in a future release, the ATMARPSV statement will no longer be supported.

Use the ATMARPSV statement to designate the ATMARP server to resolve ATMARP requests for a logical IP subnet (LIS).

Restriction: Statements describing ATM devices must be coded in the following order:

1. ATMLIS
2. DEVICE
3. LINK
4. ATMPVC (if used)
5. ATMARPSV

When an ATM device is started, TCP/IP attempts to establish a connection to the ATMARP server for any LINK associated with a device that both specifies an ATMLIS and has a corresponding ATMARPSV defined.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──ATMARPSV──arpsrv_name──lis_name──┬─SVC──ip_addr──NSAP──physical_addr─┬──────►◄
                                     └─PVC──pvc_name─────────────────────┘
```

Parameters

`arpsrv_name`
: The ATMARP server to resolve ARP requests for this LIS. An *arpsrv_name* has a maximum length of 16 characters.

`lis_name`
: The logical IP subnet (LIS) as defined previously on the ATMLIS statement and as included on the LINK statement. An *lis_name* has a maximum length of 16 characters.

SVC
Indicates that TCP/IP should connect to the ATMARP server by way of a switched virtual circuit (SVC).

ip_addr
The IP address of the ATMARP server.

Requirement: This IP address must be contained within the subnet defined by the *lis_name* parameter.

NSAP
The type of physical address; Network Services Access Point.

physical_addr
The physical address of the ATMARP server. Specify a 40-digit hexadecimal value.

Requirement: This is required only if the connection to the ATM ARP server is a switched virtual circuit (SVC).

PVC
Indicates that TCP/IP should connect to the ATMARP server by way of a permanent virtual circuit (PVC). Not all ATMARP server products support being used as an ATMARP server over a PVC connection.

pvc_name
Use to specify the PVC name of the connection to the ATM ARP server, such as ATMPVC1.

Requirement: This is required only if the connection to the ATM ARP server is a permanent virtual circuit (PVC).

Rules: The following rules apply to this parameter:
- A PVC name has a maximum length of eight characters.
- This name must match the PVC defined for the ATM port in the ATM native settings in the OSA configuration, which might further restrict the set of valid names.

Steps for modifying

Perform the following steps to modify the ATMARPSV statement:
1. Stop the associated ATM device or devices.
2. Use the VARY TCPIP,,OBEYFILE command with a data set that contains a DELETE ATMARPSV statement.
3. Use the VARY TCPIP,,OBEYFILE command with a data set that contains the updated ATMARPSV statement.
4. Start the associated ATM device or devices.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands.

Examples

This is an example of a PVC connection to an ATMARP server:

```
ATMLIS   LIS1   9.67.100.0 255.255.255.0
DEVICE   OSA1   ATM  PORTNAME  PORT1
LINK     LINK1  ATM  OSA1  LIS LIS1
ATMPVC   PVC1   LINK1
ATMARPSV ARPSV1 LIS1 PVC PVC1
```

This is an example of an SVC connection to an ATMARP server:

```
ATMLIS   LIS1   9.67.100.0 255.255.255.0
DEVICE   OSA1   ATM  PORTNAME  PORT1
LINK     LINK1  ATM  OSA1  LIS LIS1
ATMARPSV ARPSV1 LIS1 SVC 9.67.100.10
NSAP 1234567890123456789012345678901234567890
```

Related topics

- "ATMLIS statement"
- "ATMPVC statement" on page 22
- "DEVICE and LINK — ATM devices statement" on page 52
- "DELETE statement" on page 43

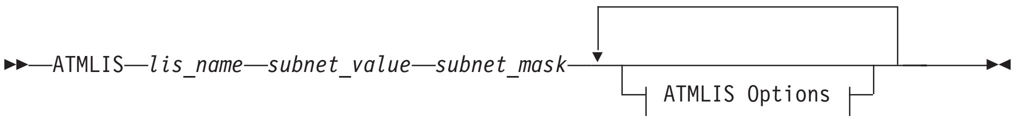
ATMLIS statement

Because support will be eliminated for the ATM device type in a future release, the ATMLIS statement will no longer be supported.

Use the ATMLIS statement to describe the characteristics of an ATM logical IP subnet (LIS). An LIS is a separate administrative ATM entity. Each logical IP subnet operates and communicates independently of other logical IP subnets on the same ATM network.

Rule: Specify the required parameters in the order shown here. The ATMLIS options can be specified in any order.

Syntax



ATMLIS Options:

```
├──┬───────────────────────────────────────┬───────────────────────────────────────────────┤
   │ ┌─ARPRETRies 2────────────────┐       │
   ├─┴─ARPRETRies──arp_retries─────┴───────┤
   │ ┌─ARPTO 3───────────────┐             │
   ├─┴─ARPTO──arp_timeout────┴─────────────┤
   │ ┌─BEARERclass C──────────┐            │
   ├─┴─BEARERclass ──class────┴────────────┤
   │ ┌─CEAGE 900───────────────────┐       │
   ├─┴─CEAGE──cache_entry_age──────┴───────┤
   │ ┌─DFLTMTU 9180──────────┐             │
   ├─┴─DFLTMTU──default_mtu──┴─────────────┤
   │ ┌─INACTVTO 300────────────────────┐   │
   ├─┴─INACTVTO──inactivity_timeout──┴─────┤
   │ ┌─MINHold 60─────────────────┐        │
   ├─┴─MINHold──min_holding_time──┴────────┤
   │ ┌─PEAKCR 0───────────────────┐        │
   └─┴─PEAKCR──peak_cell_rate─────┴────────┘
```

Parameters

lis_name
The ATM logical IP subnet on the LINK statement. A *lis_name* has a maximum length of 16 characters.

subnet_value
The subnet value that defines this logical IP subnet.

Rules:

- The *subnet_value* must be in the subnet mask. In other words, any bit in the subnet value that is a 1-bit also must be a 1-bit in the subnet mask.
- The *subnet_value* must be a class A, B, or C address.

subnet_mask
The subnet mask that defines this logical IP subnet.

ARPRETRIES *arp_retries*
The number of times an ATMARP request is tried again when no response is received and the *arp_timeout* expires. By default, two retries occur. The minimum value for this parameter is 0 and the maximum is 10. The default is 2.

ARPTO *arp_timeout*
The number of seconds to wait before retransmitting an ATMARP request. By default, the wait is 3 seconds. The minimum value for this variable is 1 second and the maximum is 60 seconds. The default is 3.

BEARERCLASS *class*
The class used to initialize the ATM session. The class is a single letter, A, C, or X. C is the default value.

CEAGE *cache_entry_age*
The number of seconds before an ARP cache entry is removed from the cache. The minimum value for this parameter is 60. The maximum and default value is 900.

DFLTMTU *default_mtu*
The maximum transmission unit for SVCs within this logical IP subnet. The minimum valid value for this parameter is 0, the maximum is 65535, and the default is 9 180.

INACTVTO *inactivity_timeout*
The number of seconds before an established SVC connection is dropped due to no traffic. A value of 0 (minimum) for this parameter indicates there is no timeout period. If a value of 1 - 9 is specified, a value of 10 is used. The maximum value is 65535, and the default is 300.

MINHOLD *min_holding_time*
The minimum number of seconds that a call remains open. A value of 0 (minimum) for this parameter indicates that the call is controlled completely by the *inactivity_timeout*. The maximum value for this parameter is 65535 and the default is 60.

Restriction: If *min_holding_time* is less than *inactivity_timeout* or if *inactivity_timeout* out is 0, then the value for *min_holding_time*has no effect.

PEAKCR *peak_cell_rate*
Indicates the best effort peak cell rate for both forward and backward traffic. A value of 0 (the minimum) indicates that a peak cell rate equal to 10% of the actual link speed is used. This is the default value. The maximum value for this variable is 2 147 483 647.

Steps for modifying

The *lis_name*, *subnet_value*, and *subnet_mask* values are used to identify each ATMLIS statement. ATMLIS options can be updated by issuing an ATMLIS statement for an existing ATMLIS with identical *lis_name*, *subnet_value*, and *subnet_mask* values. If a previously defined LIS name is used on another ATMLIS statement with a different subnet mask or subnet value, an error message is issued saying that the ATMLIS statement is already defined.

To change any options (other than subnet value and subnet mask) on the ATMLIS statement, use the VARY TCPIP,,OBEYFILE command with a data set that contains the updated ATMLIS statement. Any options not included on the ATMLIS statement are reset to defaults.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands.

Tip: The new ATMLIS values do not apply to any open ATM SVCs, but they do apply to any newly created ATM SVCs.

Perform the following steps to modify the ATMLIS statement:
1. Stop the associated ATM device or devices.

2. Use the VARY TCPIP,,OBEYFILE command with a data set that contains a DELETE ATMLIS statement and a DELETE LINK statement for each associated ATM link and a DELETE ATMARPSV statement for any associated ATMARPSV.

3. Use the VARY TCPIP,,OBEYFILE command with a data set that contains the updated ATMLIS statement along with the associated ATM LINK and ATMARPSV statements.

4. Start the associated ATM device or devices.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands.

Examples

```
ATMLIS   LIS1   9.67.100.0  255.255.255.0
```

Usage notes

- An ATMLIS must be referenced by a LINK statement. If an ATMLIS is unreferenced by any LINK statement, that ATMLIS and any ATMARPSV referring to that ATMLIS are automatically deleted.
- A HOME address used by an ATM LINK referencing an ATMLIS should be within the logical IP subnetwork defined by the LIS subnet_value and subnet_mask. If it is not within the subnetwork, the LINK is not able to send or receive data over SVCs.

Related topics

- “ATMARPSV statement” on page 17
- “DELETE statement” on page 43
- “DEVICE and LINK — ATM devices statement” on page 52
- “HOME statement” on page 136

ATMPVC statement

| Because support will be eliminated for the ATM device type in a future release, the
| ATMPVC statement will no longer be supported.

Use the ATMPVC statement to describe a permanent virtual circuit (PVC) to be used by an ATM link.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──ATMPVC──pvc_name──link_name──────────────────────────────────────────►◄
```

Parameters

pvc_name
The name of the permanent virtual circuit on the ATM network.

Requirement: This name must match the name of the PVC defined in the Open Systems Adapter (OSA) configuration in the ATM native settings for the ATM port. A *pvc_name* has a maximum length of eight characters. Because this name must match the PVC defined for the ATM port in the ATM native settings in the OSA configuration, it might further restrict the set of valid names.

link_name
The name of the ATM link associated with this PVC.

Requirement: The *link_name* must be defined previously with a LINK statement. The maximum length is 16 characters.

Steps for modifying

Perform the following steps to modify the ATMPVC statement:

1. Stop the associated ATM device whose link is referenced on the ATMPVC statement.

2. Use the VARY TCPIP,,OBEYFILE command with a data set that contains a DELETE ATMPVC statement.

3. Use the VARY TCPIP,,OBEYFILE command with a data set that contains the updated ATMPVC statement.

4. Start the associated ATM device.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands.

Examples

```
DEVICE   OSA1   ATM  PORTNAME  PORT1
LINK     LINK1  ATM  OSA1
ATMPVC   PVC1   LINK1
```

Usage notes

When an ATM device is started, TCP/IP attempts to activate all PVCs defined to all LINKs associated with the ATM device.

Related topics

- “DELETE statement” on page 43
- “DEVICE and LINK — ATM devices statement” on page 52

AUTOLOG statement

Use the AUTOLOG statement to provide a list of MVS started procedures to be started by the Autolog task when TCP/IP is started.

In addition to initially starting these procedures, the AUTOLOG statement can provide a monitoring function that ensures that these started procedures are still active. To request this monitoring function for a started procedure, reserve one or more ports for the procedure using the PORT or PORTRANGE profile statement. Do not specify the NOAUTOLOG parameter. The *proc_name* or JOBNAME value on the AUTOLOG statement entry must match the *jobname* value on the port reservation statement. Every 5 minutes, the autolog monitoring function ensures that there is either a TCP listening socket, or a UDP socket, active for those port reservations where:

- NOAUTOLOG was not specified
- The *jobname* matches the AUTOLOG statement *proc_name* or JOBNAME value

If no active socket is found for any of the reserved ports for the started procedure, then the Autolog monitoring function performs the following actions:

- Determines if the started procedure address space is still active. If it is still active, the autolog function cancels the started procedure.

  Note: If the started procedure has multiple reserved ports (for example, INETD1) and if any one of those ports does not have an active socket, the autolog function will cancel the started procedure. This can cause disruption to the active sockets for the other reserved ports for that started procedure address space.
- Restarts the started procedure.

Guideline: Ensure that ports that are used by the started procedure (for example, in /etc/services or specified on an optional port parameter for the started procedure) match the reserved ports in the port reservation statement. A mismatched port can cause the autolog monitoring function to cancel the started procedure.

Restriction: Do not use AUTOLOG to automatically start generic servers (those without affinity to a specific stack, such as TN3270E and FTP) when multiple stacks (CINET) are running. Do not use AUTOLOG to automatically start servers defined as non-cancelable (such as TN3270E) in the program properties table (PPT). Instead, use a method other than AUTOLOG to automatically start generic servers. For more information about generic servers, see z/OS Communications Server: IP Configuration Guide.

Syntax

Rule: Specify the parameters in the order shown. The optional parameters following the *proc_name* parameter can be specified in any order.

```
                        ┌──────────────────────────┐
             ┌─5────┐   ▼                          │
►►──AUTOLog──┼──────┼──proc_name──┤ Options ├──────┴──ENDAUTOLOG──────────────►◄
             └─wait─┘
```

Options:

```
├──┬─────────────────────────────┬──┬─────────────────────┬──────────────────────►
   └─PARMSTRING " parm_string"───┘  └─JOBNAME job_name────┘

►──┬───────────────────────────────┬─────────────────────────────────────────────┤
   │             ┌─────────────┐   │
   │             ▼ ┌─DVIPA─┐   │   │
   └─DELAYSTART────┼───────┼───┴───┘
                   └─TTLS──┘
```

Parameters

wait

The time TCP/IP should allow for a procedure to come down when, at startup, it is still active and TCP/IP is attempting to AUTOLOG the procedure again. This could happen if the procedure did not come down when TCP/IP was last shut down.

The default is 5 minutes. *wait* can be set to any value from 0 to 30 minutes. If a *wait* value outside the valid range is specified, the default of 5 minutes is used. When a *wait* value of 0 is specified, TCP/IP startup does not cancel and restart any procedures in the autolog list that are already started.

TCP/IP does not cancel the procedure at initialization. TCP/IP checks every 10 seconds (until the time interval specified by wait has expired) to check if the procedure has come down. If the procedure comes down during one of these 10 second intervals, it is restarted. If the procedure is still active when the time interval specified by wait expires, then TCP/IP cancels and restarts the procedure.

This value is only used at startup of TCP/IP and is never referenced again.

proc_name

A procedure that the TCP/IP address space should start.

Requirement: The procedure name must be a member of a cataloged procedure library.

PARMSTRING *"parm_string"*

A string to be added following the START *procedure_name*. Do not include the comma. The *"parm_string"* is 115 characters or less, not counting the double quotation marks around the string.

Restriction: The entire *"parm_string"* must be on one line and must not contain a double quotation mark.

JOBNAME *job_name*

The job name used for the PORT reservation statement. This can be identical to the *proc_name*, but for z/OS UNIX jobs that spawn listener threads it is not. If the *job_name* is not explicitly set, it is assumed to be the same as the *proc_name*.

DELAYSTART

An optional keyword that indicates that the procedure does not start until the TCP/IP stack has completed one or more processing steps. One or more optional subparameters determine which processing steps must be completed before the procedure is started. If no additional subparameters are configured, then the procedure is started after the TCP/IP stack has joined the sysplex group and has processed its dynamic VIPA configuration.

If this keyword is not specified, the procedure is started after the TCP/IP stack is started, whether or not the stack has completed any of the processing steps.

DVIPA

When this subparameter is specified, the procedure starts after the TCP/IP stack has joined the sysplex group and has processed its dynamic VIPA configuration. This is the default setting that occurs if DELAYSTART is coded without any subparameters.

Guideline: Use this subparameter to delay the start of a procedure that binds to a dynamic VIPA address that is created during TCP/IP stack initialization or when the procedure performs the bind. Dynamic VIPAs cannot be created until after the stack has joined the sysplex group and has processed its dynamic VIPA configuration; this keyword prevents the procedure from starting before the dynamic VIPA can be created. For information about when the TCP/IP stack joins the sysplex group, see sysplex problem detection and recovery in z/OS Communications Server: IP Configuration Guide.

Tip: The stack issues console message `EZD1214I INITIAL DYNAMIC VIPA PROCESSING HAS COMPLETED FOR` *`jobname`* when dynamic VIPA configuration processing is complete. After this console message is displayed, the autolog procedures waiting on this processing start.

TTLS When this subparameter is specified, the procedure starts after the Policy Agent has successfully installed the AT-TLS policy in the TCP/IP stack and AT-TLS services are available.

Guideline: Use this subparameter to delay the start of the procedures that depend on AT-TLS services.

Tip: The message `EZZ4250I AT-TLS SERVICES ARE AVAILABLE FOR` *`jobname`* is issued after the Policy Agent has installed the policy and the AT-TLS services are available. After this console message is issued, the autolog procedures waiting on this processing start.

Rules:

- Do not specify the DELAYSTART DVIPA (or DELAYSTART with no subparameters) for your OMPROUTE procedure if you configure the DELAYJOIN parameter on the GLOBALCONFIG profile statement.
- If TCPCONFIG TTLS is not specified in the initial profile, the DELAYSTART TTLS subparameter is ignored because AT-TLS services are not being used.

Results:

- When more than one DELAYSTART subparameter is specified, all of the processing steps defined for those subparameters must complete before the procedure is started.
- When at least one DELAYSTART subparameter is specified, but DVIPA is not specified, the default behavior does not occur; the procedure does not wait for dynamic VIPA configuration processing to complete before starting.

ENDAUTOLOG
The ENDAUTOLOG statement specifies the end of the AUTOLOG parameters to pass to TCP/IP.

Steps for modifying

To modify the AUTOLOG statement, use the VARY TCPIP,,OBEYFILE command with a data set that contains a new AUTOLOG statement. The first AUTOLOG statement in the data set replaces all previous AUTOLOG statements. Subsequent AUTOLOG statements in the same data set append to the previous statements in the data set.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands .

Examples

This example shows how to include several servers in the AUTOLOG statement:

```
AUTOLOG
   FTPD JOBNAME FTPD1    ; FTP Server
   LPSERVE               ; LPD Server
```

```
    NCPROUT                ; NCPRoute Server
    PORTMAP JOBNAME PORTMAP1      ; USS Portmap server (SUN 4.0)
    RXSERVE                ; Remote Execution Server
    SMTP                   ; SMTP Server
    OSNMPD                 ; SNMP Agent Server
    SNMPQE                 ; SNMP Client Address space
    TCPIPX25               ; X25
    MVSNFS                 ; Network File System Server
ENDAUTOLOG
```

The next example shows how to autolog two procedures using the PARMSTRING, DELAYSTART, and JOBNAME options.

- The first procedure is named MYPROC1. This procedure does not start until after the TCP/IP stack has joined the sysplex group and has processed its dynamic VIPA configuration. When the procedure is started, it should use the following MVS console start command:
  ```
  S MYPROC1,PARMS='-w 100',ID=XYZ
  ```
- The second procedure has a listening z/OS UNIX thread that is the first spawned task. (You can use the MVS DISPLAY ACTIVE,LIST console command to determine the job name.) If the MYPROC21 procedure abends or stops listening, the following MVS console start command is entered:
  ```
  S MYPROC21,PARMS='-dzy 50',DSN='HLQ.'
  ```
- The third procedure is named MYPROC3. This procedure does not start until AT-TLS services are available.
  ```
  AUTOLOG 20
  MYPROC1 PARMSTRING "PARMS='-w 100',ID=XYZ"  DELAYSTART
  MYPROC2 PARMSTRING "PARMS='-dzy 50',DSN='HLQ.'" JOBNAME MYPROC21
  MYPROC3 DELAYSTART TTLS
  ENDAUTOLOG

  PORT 2010 TCP MYPROC1
       2011 TCP MYPROC21
       2012 TCP MYPROC3
  ```

Usage notes

The AUTOLOG statement can be used to start both socket and non-socket applications. For any procedure that has no port reserved in the PORT statement, AUTOLOG initially starts the procedure when TCP/IP starts. For procedures whose ports are reserved in the PORT statement (and do not have the NOAUTOLOG option specified), each port is checked to make sure that the procedure has an active connection to that port. If a procedure has multiple ports reserved and any one port is inactive, the procedure is canceled and restarted. For TCP connections, the procedure must have a socket open to that port in the LISTEN state. For UDP connections, the procedure must have a socket open to that port.

Related topics

- “PORT statement” on page 257
- “PORTRANGE statement” on page 266

BEGINROUTES statement

Use the BEGINROUTES statement to add static routes to the main route table. The BEGINROUTES statement is an alternative to the GATEWAY statement. You can specify a BSD style syntax or destination IP address and address mask; you can also define the route to be replaceable and define IPv6 static routes.

To configure policy-based route tables, use the RouteTablestatement. For more information, see the policy-based routing information in the z/OS Communications Server: IP Configuration Guide.

IBM Health Checker for z/OS can be used to check whether the total number of indirect routes in a TCP/IP stack routing table exceeds a maximum threshold. When this threshold is exceeded, OMPROUTE and the TCP/IP stack may potentially experience high CPU consumption from routing changes. A large routing table is considered to be inefficient in network design and operation. For more details about IBM Health Checker for z/OS, see IBM Health Checker for z/OS: User's Guide..

The destination IP address can be an IPv4 or IPv6 address and does not need to be a fully qualified address.

Requirement: The first hop gateway IP address can also support either IPv4 or IPv6 addresses, but must be a fully qualified address.

Because it is compatible with UNIX standards, easier to code than GATEWAY, and has enhanced functionality, BEGINROUTES is the preferred method for defining static routes. Future static route enhancements are only available with the BEGINROUTES statement.

The IP static routes can be modified by the following methods:
- Replace the routing table using the VARY TCPIP,,OBEYFILE command.
- Incoming ICMP Redirect packets can replace IPv4 static routes, and also add routes to the routing table.
- Incoming ICMPv6 Redirect packets can replace IPv6 static routes, and also add routes to the routing table.
- Dynamic routing daemons (for example, OMPROUTE) can replace IPv4 or IPv6 replaceable static routes, as well as add dynamic routes to the routing table.
- Router advertisements can update IPv6 replaceable static routes, as well as add dynamic routes to the routing table.

The first BEGINROUTES statement of each configuration data set that is issued replaces all static routes in the existing routing table with the new route information. All static routes are deleted, along with all routes learned by way of ICMP or ICMPv6 redirects. Routes created by OMPROUTE and router advertisements are not deleted. Subsequent BEGINROUTES statements in the same data set add entries to the routing table.

Restrictions:
- A BEGINROUTES-ENDROUTES block and a GATEWAY statement cannot be intermixed in the same configuration data set. If they are intermixed, the first type found is used and the other type is discarded with warning messages being

issued to the console. You can use a BEGINROUTES-ENDROUTES block in the initial profile and a GATEWAY statement in a later VARY TCPIP,,OBEYFILE command data set, and vice versa.
- Static routes defined by the BEGINROUTES-ENDROUTES block cannot be replaced by OMPROUTE or router advertisements unless the static routes are defined as replaceable. If you want OMPROUTE or router advertisements to begin managing all routes, an empty BEGINROUTES-ENDROUTES block can be used in a VARY TCPIP,,OBEYFILE command data set to eliminate the existing static routes.
- ROUTE entries within a BEGINROUTES-ENDROUTES block can be coded only for LINK names or INTERFACE names that exist when the entry is processed.

Result: When an incorrect ROUTE entry statement is encountered, the ROUTE entry is rejected with an error message, but the rest of the ROUTE entries in that BEGINROUTES-ENDROUTES block are still processed. Subsequent BEGINROUTES-ENDROUTES blocks in the same initial profile or VARY TCPIP,,OBEYFILE command data set, are also processed.

Route precedence is as follows:
- If a route exists to the destination address (a host route), it is chosen first.
- For IPv4, if subnet, network, or supernetwork routes exist to the destination, the route with the most specific network mask (the mask with the most bits on) is chosen second.
- For IPv6, if prefix routes exist to the destination, the route with the most specific prefix is chosen second.
- If the destination is a multicast destination and multicast default routes exist (valid only for IPv4), the route with the most specific multicast address is chosen third.
- Default routes are chosen when no other route exists to a destination.

Rule: The required parameters for this statement must be specified in the order shown here. The optional parameters can be specified in any order.

Syntax

```
                  ┌─────────────────┐
                  ▼                 │
►►──BEGINRoutes────┤ Route Entry ├───┴──ENDRoutes──────────────────────────────►◄
```

Route Entry:

```
   ┌─────────────────────────────────────────────────────────────────────────────────┐
   ▼                                                                    ┌─────────┐    │
├────ROUTE──┤ Destination ├──┬─gateway_addr─┬──interface_name──┤ PacketSize ├──▼─────────┴──┴────┤
                            └─=────────────┘                                    └─Options─┘
```

Destination:

```
├──┬──┤ IPv4_Destination ├──┬──────────────────────────────────────────────────┤
   └──┤ IPv6_Destination ├──┘
```

IPv4_Destination:

```
├──┬─dest_ipaddr address_mask──┬────────────────────────────────────────────┤
   ├─dest_ipaddr HOST──────────┤
   ├─dest_ipaddr/num_mask_bits─┤
   └─DEFAULT───────────────────┘
```

IPv6_Destination:

```
├──┬─DEFAULT6─────────────────┬─────────────────────────────────────────────┤
   ├─dest_ipaddr HOST─────────┤
   └─dest_ipaddr/prefixLength─┘
```

Packet size:

```
├──MTU──┬─mtu_size────┬─────────────────────────────────────────────────────┤
        └─DEFAULTSIZE─┘
```

Options:

```
   ┌─DELAYAcks───┐   ┌─MAXImumretransmittime 120.00──────┐
├──┼─────────────┼───┼───────────────────────────────────┼──────────────────►
   └─NODELAYAcks─┘   └─MAXImumretransmittime ──seconds───┘

   ┌─MINImumretransmittime 0.50──────┐   ┌─NOREPLaceable─┐
►──┼─────────────────────────────────┼───┼───────────────┼──────────────────►
   └─MINImumretransmittime ──seconds─┘   └─REPLaceable───┘

   ┌─ROUNDTRIPGain 0.125───┐   ┌─VARIANCEGain 0.25─────┐
►──┼───────────────────────┼───┼───────────────────────┼────────────────────►
   └─ROUNDTRIPGain ──value─┘   └─VARIANCEGain ──value──┘

   ┌─VARIANCEMultiplier 2.00─────┐
►──┼─────────────────────────────┼──────────────────────────────────────────┤
   └─VARIANCEMultiplier ──value──┘
```

Parameters

dest_ipaddr
: The destination IPv4 or IPv6 address.

  The DEFAULT/DEFAULT6 keyword in this field specifies default routes. For IPv4, the destination address can be a host, network, subnetwork, supernetwork or default address. For IPv6, the destination address can be a host, prefix or default address. In addition, multiple routes having an identical destination IP address and address mask can be specified. When multiple routes are specified, all of them are used when multipath is enabled on the IPCONFIG/IPCONFIG6 statement; otherwise, only the first active route specified is used.

  Requirement: An IPv4 address must be fully qualified.

address_mask
: The BSD style address mask for an IPv4 route. If the HOST keyword is specified in this field, it is a host route with a mask of 255.255.255.255.

num_mask_bits
An integer value in the range 1 - 32 that represents the number of leftmost significant bits for the address mask of an IPv4 route.

prefixLength
An integer value in the range 1 - 128 representing the number of bits in the *dest_ipaddr* value that are used to determine the destination address of the IPv6 route.

gateway_addr
The host IPv4 or IPv6 address of a gateway or router that you can reach directly, and that forwards packets for the destination network or host over the interface that is identified by *interface_name*.

Requirement: This value must be either a fully qualified address or an equal sign (=), meaning that the messages are routed directly to destinations on that network or directly to that host. The equal sign is not supported for DEFAULT or DEFAULT6 route entries. It cannot be a local IP address on this TCP/IP stack. A local IP address can be defined on the HOME, INTERFACE, VIPADEFINE, or IPCONFIG/IPCONFIG6 DYNAMICXCF statement.

interface_name
The name of the interface through which packets are sent to the specified destination.

Requirement: The interface name must be previously defined in a LINK or INTERFACE statement. VIPA interfaces are not allowed on the ROUTE entry statement.

MTU *mtu_size*
The maximum transmission unit (MTU) in bytes for the destination. This value can be up to 65535. The keyword DEFAULTSIZE in this field requests that TCP/IP supply a default value of 576 for IPv4 routes and 1280 for IPv6 routes.

See Figure 1 on page 48 for more information about the largest MTU value supported by each IPv4 link type.

See Table 6 on page 142 for more information about the largest MTU value supported by each IPv6 interface type.

See the Usage notes in this topic for packet size considerations.

Tip: See z/OS Communications Server: IP Configuration Guide, under section, Maximum transmission unit considerations, for additional information about how TCP/IP uses the MTU to determine the largest size frame to send.

REPLACEABLE | NOREPLACEABLE
Indicates whether or not the static route can be replaced by OMPROUTE and router advertisements when a dynamic route to the same destination is discovered.

NOREPLACEABLE
Indicates that static routes cannot be replaced by dynamic routes. The static route is always used to reach the destination, regardless of when dynamic routes are available. This is the default setting. This parameter can be abbreviated as NOREPL.

REPLACEABLE
Indicates that the static route can be replaced by OMPROUTE and router advertisements when a dynamic route to the same destination is discovered. This parameter can be abbreviated REPL.

Restrictions:

- Only one type (replaceable or nonreplaceable) of static route can be defined to the same destination. All static routes defined to a destination must match the type of the first static route defined to that destination. Any definitions that do not match that type are rejected.
- Replaceable static routes cannot be defined to destination addresses that correspond to dynamic VIPAs for which the TCP/IP stack is a sysplex distributor target.

Tip: You can use the Netstat ROUTE/**-r** RSTAT command to display all replaceable static routes currently configured.

Retransmission parameter considerations

The parameters listed in this topic affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a specified number of retransmits, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times that packets are retransmitted before the connection times out differs for initial connection establishment and for data packets . For initial connection establishment, the initial time interval is set at approximately 3 seconds and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval, and data packets are retransmitted 15 times before the connection is timed out. All of the remaining parameters listed in this topic affect the data packet retransmission algorithm. Only the MINIMUMRETRANSMITTIME parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

The retransmission parameters enable system administrators who are familiar with TCP/IP transmission performance to alter the flow of TCP/IP data packets and acknowledgments. Under normal circumstances, the following occurs:
- TCP typically waits to receive two packets before sending one ACK to acknowledge the data within them.
- When TCP sends a packet, it waits for an acknowledgment. If it times out before getting an acknowledgment, it resends the packet.

Use the following parameters to adjust the retransmission time-out calculations; slower transmission times prevent packets from being resent as quickly:
- MAXIMUMRETRANSMITTIME
- MINIMUMRETRANSMITTIME
- ROUNDTRIPGAIN
- VARIANCEGAIN
- VARIANCEMULTIPLIER
- DELAYACKS

- NODELAYACKS

TCP uses these values in an algorithm called the TCP Retransmission Timeout Calculation, which is described in RFC 793. When you use this calculation, the following occurs:

- A smoothed round trip time (SRTT) and variance (VAR) is updated from the individual RTT derived from each packet acknowledgement.
- The retransmit time for a new packet is set to twice (approximately) the current SRTT value plus the VAR value.
- Each time a packet is retransmitted, the retransmit time value is doubled.
- The actual interval time used for the initial packet and each retransmission is the retransmit time calculated previously, but limited by the configured MINIMUMRETRANSMITTIME and MAXIMUMRETRANSMITTIME values.

DELAYACKS | NODELAYACKS

Controls transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header.

NODELAYACKS

Specifies that an acknowledgment is returned immediately when a packet is received with the PUSH bit on in the TCP header. The NODELAYACKS parameter on the BEGINROUTES, GATEWAY, and RouteTable statements affects only the connections that use this route. Specifying NODELAYACKS on the TCP/IP stack BEGINROUTES or GATEWAY profile statements, or on the Policy Agent RouteTable statement, overrides the specification of the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

DELAYACKS

Delays transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. The DELAYACKS parameter on the BEGINROUTES, GATEWAY, and RouteTable statements affects only the connections that use this route. This is the default, but you can override the default by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements.

MAXIMUMRETRANSMITTIME

Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to timeout. Specifying MAXIMUMRETRANSMITTIME assures that the interval time never exceeds the specified limit. The minimum value that can be specified for MAXIMUMRETRANSMITTIME is 0. The maximum is 999.990. The default is 120 seconds. This parameter does not affect initial connection retransmission.

MINIMUMRETRANSMITTIME

Sets a minimum retransmit interval. Increasing this value might increase the amount of time it takes for TCP to time out a connection. The minimum value that can be specified for MINIMUMRETRANSMITTIME is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

ROUNDTRIPGAIN

This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet RTT has on the average. The minimum value that can be specified for ROUNDTRIPGAIN is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

VARIANCEGAIN
This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum value that can be specified for VARIANCEGAIN is 0. The maximum value is 1.0. The default is 0.25. This parameter does not affect initial connection retransmission.

VARIANCEMULTIPLIER
This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT has on calculating the retransmission interval. The minimum value that can be specified for VARIANCEMULTIPLIER is 0. The maximum value is 99.990. The default is 2. This parameter does not affect initial connection retransmission.

Retransmission parameters

Use the ROUNDTRIPGAIN, VARIANCEGAIN, and VARIANCEMULTIPLIER parameters to instruct TCP how heavily to weigh the most recent behavior of the network versus the long term behavior for updating the SRTT and VAR values. If you specify smaller values for these parameters, TCP attempts to correct for congestion only if the congestion is sustained. With larger values, TCP corrects for congestion more quickly, and the system is more sensitive to variations in network performance. Use the default values (unless your retransmission rate is too high).

Use DELAYACKS to delay the acknowledgments so that they can be combined with data sent to the foreign host.

Steps for modifying

To modify any values on the BEGINROUTES-ENDROUTES block, use a VARY TCPIP,,OBEYFILE command with a data set that contains a new BEGINROUTES-ENDROUTES block. All existing static routes are deleted, along with all routes learned by way of ICMP or ICMPv6 redirects. Routes created by OMPROUTE and router advertisements are not deleted. To remove all static routes from the main routing table, specify an empty BEGINROUTES-ENDROUTES block.

Results:
- If any HOME list entries are changed or deleted, all static routes using the associated links are deleted. This applies to IPv4 only.
- If any INTERFACE statements are deleted, all static routes that correspond with the INTERFACE names are deleted.
- If a LINK or INTERFACE becomes inactive, then all routes that are associated with that link or INTERFACE are marked inactive.
- If a LINK or INTERFACE becomes active, then all static routes that are associated with that link or INTERFACE are marked active.

Examples

```
; BEGINRoutes: Defines static routes to the main route table for IPv4
;                  and IPv6
;
BEGINRoutes
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
```

```
;      Destination Subnet Mask   First Hop    Link Name Packet Size
;
 ROUTE 130.50.75.0 255.255.255.0 =            TR1       MTU 2000
 ROUTE 193.5.2.0/24              =            ETH1      MTU 1500
 ROUTE 9.67.43.0   255.255.255.0 =            FDDI1     MTU 4000
 ROUTE 193.7.2.2   HOST          =            SNA1      MTU 2000
;
;      Destination Subnet Mask   First Hop  Interface   Packet Size
;
 ROUTE fe80::230:71ff:fed3:5160  HOST =        OSAQDIO26     MTU 2000
 ROUTE 2001:0CD8:1/128                =        OSAQDIO26     MTU 2000
;
;
; Indirect Routes - Routes that are reachable through routers on my
;                  network.
;
;      Destination Subnet Mask   First Hop    Link Name Packet Size
;
 ROUTE 193.12.2.0  255.255.255.0 130.50.75.10 TR1       MTU 2000
 ROUTE 10.5.6.4    HOST          193.5.2.10   ETH1      MTU 1500
;
;      Destination Subnet Mask   First Hop  Interface   Packet Size
;
 ROUTE FEC8::/64           fe80::230:71ff:fed3:5160     OSAQDIO26   MTU 2000
;
; Default Route - All packets to an unknown destination are routed
;                through this route.
;
;      Destination               First Hop    Link Name Packet Size
;
 ROUTE DEFAULT                   9.67.43.99   FDDI1     MTU DEFAULTSIZE
;
;      Destination Subnet Mask   First Hop  Interface   Packet Size
;
 ROUTE DEFAULT6             fe80::230:71ff:fed3:5160   OSAQDIO26   MTU DEFAULTSIZE
ENDRoutes
```

Usage notes

- The destination address and first hop IP address must both be either IPv4 or IPv6. If they do not match, an error message is displayed.
- An error message is displayed if an IPv6 address is coded along with an IPv4 link name, or if an IPv4 address is coded along with an IPv6 interface name.
- If the first hop IP address is IPv6, then it cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IP address with the reserved prefix ::/96. If the IPv6 address is one of these types, an error message is displayed.
- If the destination address is an IPv4-mapped IPv6 address, an error message is displayed.
- The host portion of a valid host IP address cannot be all ones or all zeros; an address that consists of all ones or zeros is considered to be the broadcast address. The *dest_ipaddr* value can be either a network address or a host IP address. The *gateway_addr* value must be a host IP address.
- Packet size considerations:
  - The *mtu_size* value that z/OS Communications Server can handle varies for different networks. For example, while the largest packet size for the Ethernet protocol is 1500 bytes, the largest packet size for the 802.3 protocol is 1492 bytes.
  - The actual packet size is determined by the total network connection.
    - If a locally attached host has a packet size smaller than yours, transfers to that host use the smaller size.

    - The TCP maximum segment size for the 3172 Interconnect Controller Program is 4096. Any packet specifications over 4096 are limited by this restriction. For example, if you specified a packet size of 4352, the resulting packet size would still only be 4096 + the header = 4132.
  - Large packets can be fragmented by intervening gateways for IPv4 only. Fragmentation and reassembly of packets are expensive in their use of bandwidth and CPU time. Therefore, packets sent through gateways to other networks should use the default size, DEFAULTSIZE, unless one of the following situations is true:
    - All intervening gateways and networks are known to accept larger packets
    - Path MTU discovery (PATHMTUDISCOVERY) is enabled on the IPCONFIG statement, which results in the TCP/IP stack dynamically learning the maximum MTU for the total network connection. For IPv6, Path MTU discovery is always enabled.
  - If this is a pSeries link, the *mtu_size* value cannot exceed the *write_size* specified on the corresponding DEVICE statement.
  - You cannot specify an MTU smaller than the default MTU size. For IPv4 the default MTU is 576 and for IPv6 it is 1280.
- If the routing table is empty, all addresses in the HOME list remain route capable. For information about testing commands with LOOPBACK, see the z/OS Communications Server: IP User's Guide and Commands.
- The IPv4 *address_mask* value must follow the Classless Inter-Domain Routing (CIDR) convention that requires the actual mask to be one or more on-bits followed by zero or more off-bits. You cannot have on-bits followed by off-bits followed by on-bits. Therefore, a mask of 255.255.254.0 (or FFFFFE00) is valid, but a mask of 255.255.253.0 (or FFFFFD00) is not valid because 253 is 11111101.
- There is no limit on the number of equal-cost multipath routes to a destination.
- Multicast routes can be specified using host specification. You can also specify multicast network or prefix routes by using BEGINROUTES. A general multicast default route for IPv6 can be specified using:

```
BEGINROUTES
ROUTE FF00::/8 = INTERFACE1  MTU 4096
ENDROUTES
```

Related topics

- “BSDROUTINGPARMS statement”
- “GATEWAY statement” on page 109
- “IPCONFIG statement” on page 190
- “IPCONFIG6 statement” on page 206
- Policy agent Route table, see “RouteTable statement” on page 1152

BSDROUTINGPARMS statement

Restriction: The BSDROUTINGPARMS statement applies only to IPv4 interfaces defined with the LINK statement.

Use the BSDROUTINGPARMS statement to define the characteristics of every physical link defined at the host. This includes links used for static routing and links over which NCPROUTE sends transport PDUs to client NCPs.

For more information about subnet masking, see z/OS Communications Server: IP Configuration Guide.

When not using OMPROUTE, define links in BSDROUTINGPARMS. Otherwise, the values for MTU and subnet mask for links are filled in from BEGINROUTES or GATEWAY statements, if any. These assumed definitions might not provide good performance or function.

If using OMPROUTE, it is not necessary to define the BSDROUTINGPARMS statement because the parameters are overridden by OMPROUTE. However, if the Ignore_Undefined_Interfaces option is defined in OMPROUTE such that a default interface definition is not generated for a corresponding link, BSDROUTINGPARMS might need to be defined to specify interface characteristics for that link. If OMPROUTE does not have the equivalent parameters coded for a corresponding link, it provides defaults that might not provide good performance or function.

Requirement: If using NCPROUTE with OMPROUTE, the BSDROUTINGPARMS statement is required to route Transport PDUs prior to OMPROUTE activation. Because the BSDROUTINGPARMS parameters are overridden by the interface parameters defined in the OMPROUTE configuration, ensure that the interface parameters for the SNALINK or IP/CDLC channel connections are identical in both the BSDROUTINGPARMS statement and the OMPROUTE configuration file.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──BSDRoutingparms──┬─TRUE──┬──────────────────────────────────────────────►
                     └─FALSE─┘


  ┌──────────────────────────────────────────────────────────────┐
  ▼                                                              │
►───link_name──┬─DEFAULTSize─┬──cost_metric──subnet_mask──dest_addr─┴──────────►
               └─mtu─────────┘

►──ENDBSDRoutingparms──────────────────────────────────────────────────────►◄
```

Parameters

TRUE
Specifies that the maximum packet size for the interface is always used regardless of whether the destination is one or more hops away.

FALSE
Specifies that the default maximum packet size of 576 is used (rather than the packet size of the interface) when sending to networks that are not locally attached.

link_name
The name of the link as defined in a LINK statement.

Requirements:

- Each link must be defined once in the BSDROUTINGPARMS statement.

- To be used, a link must be defined at the time the BSDROUTINGPARMS statement is processed. If the corresponding link name is not defined in the HOME list, the link has no HOME address and is rendered as unusable until a HOME address is assigned.

`mtu`

The maximum packet size for this interface. The DEFAULTSIZE keyword can be used to designate the default of 576. The minimum value is 1, and the maximum value is 65 535.

See Figure 1 on page 48 for more information about the largest MTU value supported by each link type.

The MTU value specified on BSDROUTINGPARMS statement is also used for applications that use the setsockopt() IP_MULTICAST_IF option to specify the route for multicast datagrams.

Tip: See z/OS Communications Server: IP Configuration Guide, section Maximum transmission unit considerations, for additional information about how TCP/IP uses the MTU to determine the largest size frame to send.

`cost_metric`
Specifies the interface-level metric associated with the cost of use for the link. If using OMPROUTE, the value of *cost_metric* is overridden with a corresponding interface parameter value that might be coded or set (Cost0= on OSPF_INTERFACE or In_Metric= on RIP_INTERFACE). The default is 0.

`subnet_mask`
A bit mask (expressed in dotted decimal form), having bits in the network or host portions, that define the link-level subnet mask associated with the link and acts as a default for a route-level subnet mask to be used for routes dynamically created over this link.

Requirement: The bits must be contiguous from left to right.

The *subnet_mask* is related to the HOME IP address of the link. If the *subnet_mask* equals 0 to indicate that the network is not subnetted, the default is the network class mask, which is based on an IP address class. By definition, the network class masks are:
- Class A: 255.0.0.0
- Class B: 255.255.0.0
- Class C: 255.255.255.0

A subnet mask is used in calculation of a subnet, network, or supernet route. A subnet route is used to represent multiple hosts in a subnet, a network route is used to represent multiple subnet routes (or multiple hosts if the network is not subnetted), and a supernet route is used to represent multiple network routes in a supernet. For Classless Inter-Domain Routing (CIDR) support, variable-length subnet masks can be used. Variable-length subnet masks can be used in a single network; that is, multiple subnets having the same network number can have different subnet masks. Fixed-length subnet masks are used in a single network with multiple subnets having the same network number and subnet mask. A subnet mask that is less than the network class mask is considered to be a supernet mask. A supernet mask can be defined such that multiple networks can be represented by a single supernet.

Restriction: The host mask of 255.255.255.255 cannot be used for the interface-level subnet mask; however, an implicit host route based on its home IP address is dynamically created internally for this link.

dest_addr

Destination address applies to point-to-point links only. A nonzero destination address applies to nonbroadcast-capable and nonmulticast-capable point-to-point links. If 0 is coded, a directed broadcast or multicast address is used; otherwise, insert the address of the host on the other end of the link. For VIPA links, this field should be 0.

See Figure 1 on page 48 for more descriptions about devices and links.

Steps for modifying

To modify the BSDROUTINGPARMS statement for a link, use a VARY TCPIP,,OBEYFILE command with a data set which defines a new BSDROUTINGPARMS statement for a link with the same *link_name*. The new BSDROUTINGPARMS statement is a complete replacement for the original BSDROUTINGPARMS statement. If you have changed the link's IP address, or the order of the HOME list entries, along with the BSDROUTINGPARMS changes, remember to include the new HOME list statement in the same VARY TCPIP,,OBEYFILE command data set as the new BSDROUTINGPARMS statement.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands .

Table 3. BSDROUTINGPARMS modification methods

Modification method	**Required action**
Adding new links	Issue VARY TCPIP,,OBEYFILE command with new DEVICE, LINK, HOME, and BSDROUTINGPARMS statements.
Deleting or changing order of links in use.	Issue VARY TCPIP,,OBEYFILE command with new HOME statement.
Changing HOME IP addresses or BSDROUTINGPARMS values for existing links in use.	Issue VARY TCPIP,,OBEYFILE command with new HOME statements or BSDROUTINGPARMS statements, or both.
Guideline: If HOME addresses have been changed, the NCP generation definitions must also be changed in order to recognize the new HOME addresses.	

Examples

This example shows the BSDROUTINGPARMS statement for several types of LAN media.

```
; link     maxmtu   metric   subnet mask     dest addr
BSDROUTINGPARMS false
   TR1       2000      0     255.255.255.0   0
   ETH1      1500      0     255.255.255.0   0
   FDDI1  DEFAULTSIZE   0     255.255.255.0   0
ENDBSDROUTINGPARMS
```

This example includes a link, `LINK3`, that is a point-to-point link between host MVS1 and host 128.84.54.6.

```
;
; link     maxmtu    metric    subnet_mask    dest_addr
BSDROUTINGPARMS false
   LINK1  DEFAULTSIZE    0      255.255.255.0     0
   LINK2  DEFAULTSIZE    0      255.255.255.0     0
   LINK3  1500           0      255.255.255.0     128.84.54.6
ENDBSDROUTINGPARMS
```

This example shows the definitions for VIPA links.

```
BSDROUTINGPARMS false
   VLINK1   DEFAULTSIZE  0  255.255.255.252  0
   VLINK2   DEFAULTSIZE  0  255.255.255.252  0
ENDBSDROUTINGPARMS
```

This example shows how BSDRoutingparms relate to other statements in the profile.

```
    DEVICE DEVC00   CTC C00 IOBUFFERSIZE 65535 AUTORESTART
        LINK   LCTCC00  0   DEVC00 IFSPEED 10000
        HOME   9.32.2.1 LCTCC00

    DEVICE DEVD00   LCS D00 AUTORESTART
        LINK   ETHERD00  ETHERNET 0 DEVD00
        LINK   IBMTRD00  IBMTR    1 DEVD00
        LINK   FIDDID00  FDDI     2 DEVD00
        HOME   130.80.0.1 ETHERD00
               130.81.0.2 IBMTRD00
               130.82.0.3 FIDDID00

    PRIMARYINTERFACE LCTCC00

    BSDROUTINGPARMS TRUE
       LCTCC00   DEFAULTSIZE 0  255.252.0.0   9.32.2.5
       ETHERD00  DEFAULTSIZE 0  255.252.0.0   0
       IBMTRD00  DEFAULTSIZE 0  255.252.0.0   0
       FIDDID 00 DEFAULTSIZE 0  255.252.0.0   0
    ENDBSDROUTINGPARMS


START  DEVD00
```

This example shows how to use BSDRoutingparms with supernet routes.

```
 HOME
    130.201.1.1  VLINK1
    172.200.10.1 ETH1
    192.3.200.1  CTCBF0

 BSDROUTINGPARMS FALSE
    ETH1   1500    0       255.252.0.0      0
    VLINK1 1500    0       255.254.0.0      0
    CTCBF0 1000    0       255.255.252.0    192.3.200.2
ENDBSDROUTINGPARMS
```

Usage notes

- For rules on defining virtual IP addresses for VIPA links, see "HOME statement" on page 136.
- The maximum transmission unit (MTU) and metric of any other links with a destination address in the same subnet are updated to ensure that all entries in the same subnet have the same routing values. Except for these links and the LOOPBACK link, all links get default BSD values if not specified.
- If no HOME address exists for a LINK or if a HOME address is changed by way of a later VARY TCPIP,,OBEYFILE command, processing of the HOME statement verifies whether *subnet_mask* value on the BSDROUTINGPARMS statement is within the valid ranges.
- When an incorrect BSDROUTINGPARMS entry is encountered, all entries following that entry on that BSDROUTINGPARMS statement are ignored. Subsequent BSDROUTINGPARMS statements are processed.
- BSDROUTINGPARMS statements can only be coded for LINK names that exist in the HOME list when the statement is processed. Thus, LINKs from

IPCONFIG DYNAMICXCF and the VIPADYNAMIC block should not be included in BSDROUTINGPARMS statements of the initial PROFILE.TCPIP. However, the data set used on a VARY TCPIP,,OBEYFILE command can contain BSDROUTINGPARMS statements with LINKs from IPCONFIG DYNAMICXCF and the VIPADYNAMIC block.

- The BSDROUTINGPARMS parameter values are displayed with the Netstat DEvlinks/**-d** commands. If the BSDROUTINGPARMS statement is not defined, the values of the displayed parameters are either the defaults from the BEGINROUTES or GATEWAY statement, or are from the OMPROUTE configuration statements.

Related topics

- "BEGINROUTES statement" on page 28
- "DEVICE and LINK — VIRTUAL devices statement" on page 101
- "GATEWAY statement" on page 109
- "HOME statement" on page 136
- "IPCONFIG statement" on page 190

DEFADDRTABLE statement

Use the DEFADDRTABLE statement to configure the policy table for IPv6 default address selection. If you do not configure the policy table for IPv6 default address selection with the DEFADDRTABLE profile statement, then the following default policy table is used:

```
Prefix            Precedence Label
::1/128           50         0
::/0              40         1
2002::/16         30         2
::/96             20         3
::ffff:0.0.0.0/96 10         4
```

Restriction: Only one DEFADDRTABLE block should appear in a configuration data set. Any subsequent DEFADDRTABLE blocks are ignored, and an informational message is displayed. If a syntax error is encountered when this statement is processed, an error message is displayed, and the entire DEFADDRTABLE block is ignored (no entries are processed).

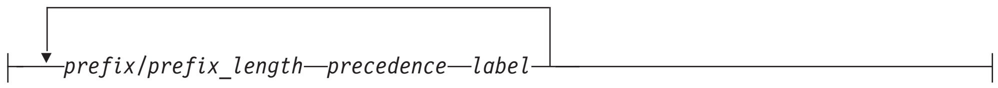
Guideline: The order of the entries in the policy table is not important. When the policy table is used during address selection, all entries in the table are searched to locate the entry with the prefix that best matches (longest prefix match) the address for which precedence and label values are needed.

For more information about the policy table for default address selection and the precedence and label values, see z/OS Communications Server: IPv6 Network and Application Design Guide.

Syntax

```
                  ┌───────────────────┐
                  ▼                   │
►►──DEFADDRTable────┤ Policy Entry ├────ENDDEFADDRTable─────────────────►◄
```

Policy Entry:



Parameters

prefix/prefix_length
: The address prefix that is used to select the policy table entry that best matches a source address or a destination address. The policy table is a longest-matching-prefix lookup table.

  If duplicate prefix entries are specified in the same DEFADDRTABLE block, the first prefix entry is used, the remaining duplicate prefix entries are ignored, and messages are displayed.

  prefix The digits (in colon-hexadecimal format) before the slash (/) specify the prefix.

  prefix_length
  An integer value in the range 0 - 128 that specifies the length of the prefix, in bits.

precedence

An integer value in the range 0 - 65530 that specifies the precedence that is used to sort destination addresses.

label

An integer value in the range 0 - 65530 that specifies that a particular source address prefix is preferred for use with a destination address prefix.

Steps for modifying

The following considerations apply when you modify the DEFADDRTABLE block:

- To remove all the current configured policies, specify the DEFADDRTABLE block without any entries:

  ```
  DEFADDRTABLE ENDDEFADDRTABLE
  ```

  Issue the VARY TCPIP,,OBEYFILE command; the default policy table replaces the previously configured values.
- To change any of the policy entries, create a DEFADDRTABLE block with the existing set of policies. Update the policy entries that need to be changed. Then, issue the VARY TCPIP,,OBEYFILE command to activate the change. The new policy table completely replaces the existing policy table.

Examples

The default policy table contains the following values:

```
DEFADDRTABLE
;   Prefix             Precedence Label
    ::1/128            50         0
    ::/0               40         1
    2002::/16          30         2
    ::/96              20         3
    ::ffff:0.0.0.0/96  10         4
ENDDEFADDRTABLE
```

This default table specifies the following behavior:

- Prefer using native source addresses with native destination addresses, 6to4 source addresses with 6to4 destination addresses, and IPv4-compatible source addresses with IPv4-compatible destination addresses

  Guideline: IPv4-compatible addresses have been deprecated by RFC 4291, but are shown here because they are part of the default policy table defined by RFC 3484.
- Prefer using IPv6 network transport over IPv4 network transport when possible

To specify that IPv4 network transport should be preferred over IPv6 network transport, change the precedence of the ::ffff:0.0.0.0/96 prefix to 100.

```
DEFADDRTABLE
;  Prefix              Precedence Label
   ::1/128             50         0
   ::/0                40         1
   2002::/16           30         2
   ::/96               20         3
   ::ffff:0.0.0.0/96   100        4
ENDDEFADDRTABLE
```

All other considerations being equal, a destination address whose label does not match the label of any of the possible source addresses prefers an IPv4 source address because the precedence value for the ::ffff:0.0.0.0/96 prefix is higher than the precedence value of all the other entries.

The destination address selection rules give preference to destinations of smaller scope. For example, a link-local destination is sorted before a global scope destination when the two are otherwise equally suitable. To sort global destinations before link-local destinations, change the policy table to reverse the existing preference.

```
DEFADDRTABLE
;  Prefix              Precedence Label
   ::1/128             50         0
   ::/0                40         1
   fe80::/10           33         1
   2002::/16           30         2
   ::/96               20         3
   ::ffff:0.0.0.0/96   100        4
ENDDEFADDRTABLE
```

DELETE statement

Use the DELETE statement to delete a previously defined ATMARPSV, ATMLIS, ATMPVC, device, link, port, or portrange.

Guideline: Use the INTERFACE statement with the DELETE parameter to delete a previously defined interface.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──DELEte──ATMARPSV──arpsrv_name──────────────────────────────────────────►◄
```

```
►►──DELEte──ATMLIS──lis_name──────────────────────────────────────────────────►◄
```

```
►►──DELEte──ATMPVC──pvc_name──────────────────────────────────────────────────►◄
```

```
►►──DELEte──DEVice──device_name───────────────────────────────────────────────►◄
```

```
►►──DELEte──LINK──link_name───────────────────────────────────────────────────►◄
```

```
                 ┌──────────────────────────────────────────────────┐
                 ▼                                                  │
►►──DELEte──PORT───┬─num────┬─TCP─┬────┬─RESERVED─────────────────────┬─┴──────────►◄
                   │        └─UDP─┘    └─jobname──┤ Port Options ├──┘ │
                   └─UNRSV────┬─TCP─┬───┬─jobname─┬───┬──────────────┬─┘
                              └─UDP─┘   └─*───────┘   └─Port options─┘
```

Port Options

The optional parameters for the PORT profile statement can be specified on the DELETE PORT statement but, though the syntax of the parameters is verified, the parameter values are ignored.

```
►►──DELEte──PORTRange─────────────────────────────────────────────────────────►


   ┌─────────────────────────────────────────────────────────────┐
   ▼                                                             │
►────1st_port──num_ports────┬─TCP─┬────┬─RESERVED──────────────────────┬─┴──────►◄
                            └─UDP─┘    ├─AUTHPORT──────────────────────┤
                                       └─jobname──┤ Portrange Options ├─┘
```

Portrange Options

The optional parameters for the PORTRANGE profile statement can be specified on the DELETE PORTRANGE statement but, though the syntax of the parameters is verified, the parameter values are ignored.

Parameters

The values of the required parameters must match the existing reservation or the delete fails. You can specify the optional parameters for the PORT or PORTRANGE profile statement on the DELETE PORT or DELETE PORTRANGE statement. However, even though the syntax of the parameters is verified, the parameter values are ignored.

The following parameters are network interface parameters:

arpsrv_name
: The name of the ATMARP server to be deleted. This is the name that was used on an ATMARPSV statement to define the ATMARP server to TCP/IP.

lis_name
The name of the LIS to be deleted. This is the name that was used on an ATMLIS statement to define the LIS to TCP/IP.

pvc_name
The name of the PVC to be deleted. This is the name that was used on an ATMPVC statement to define the PVC to TCP/IP.

device_name
The name of the device to be deleted. This is the name that was used on a DEVICE statement to define the device to TCP/IP.

link_name
The name of the link to be deleted. This is the name that was used on a LINK statement to define the link to TCP/IP.

The following parameters are PORT and PORTRANGE parameters.

To delete an existing PORT or PORTRANGE reservation, use a PORT or PORTRANGE profile statement and prefix it with the DELETE keyword. The only required parameters on the DELETE PORT or DELETE PORTRANGE statement are as follows:

- Reserved port number or UNRSV on the DELETE PORT statement, or the range of port numbers on the DELETE PORTRANGE statement
- Protocol of TCP or UDP
- Job name specification

num
The port number of the port to be deleted. This is the port number that was used on a PORT statement to define the port to TCP/IP.

UNRSV
UNRSV indicates that a statement that defines access to unreserved port numbers is to be deleted.

1st_port
The first port number in the range of reserved ports to be deleted.

num_ports
The number of ports to be deleted, starting with the port specified on the *1st_port* parameter. This range is the same number of ports that were reserved when the port range was defined with the PORTRANGE statement.

jobname
The job name associated with the port to be deleted.

RESERVED
Indicates that the port is not available for use by any user. Use this value to lock certain ports. This value is optional and valid for TCP and UDP protocols.

AUTHPORT
Indicates that the port is not available for use by any user except FTP, and only when FTP is configured to use PASSIVEDATAPORTS. AUTHPORT is valid only with the TCP protocol.

Steps for modifying

Modification is not applicable to this statement.

Statement dependency

- To delete a link, you must first delete any associated HOME entry by specifying a HOME statement that does not include the link, and you must also stop the device.

  Restriction: You do not need to (and cannot) stop the device when deleting a link for a virtual device.
- To delete an ATM link, you must first delete any associated ATMPVCs.
- To delete an ATMLIS, you must first delete all associated LINKs and ATMARPSVs.
- To delete an ATMARPSV, you must first stop all devices that have a LINK associated with the ATMLIS for the ATMARPSV.
- You can delete an ATMPVC for a started device. However, if the PVC is in use as an ATMARP server, you must first stop the devices using the PVC as an ATMARP server in order to delete the ATMPVC.
- To delete a device, you must first stop the device, then delete all associated links.

  Restriction: You do not need to (and cannot) stop the device when deleting a link for a virtual device.

Examples

This example shows DELETE statements to delete an ATM PVC named PVC1, an ATM LIS named LIS1, and an ATMARPSV named ARPSV1:

```
DELETE ATMPVC PVC1
DELETE ATMLIS LIS1
DELETE ATMARPSV ARPSV1
```

This example shows DELETE statements that delete a link called sanjose and a device called ourctc:

```
DELETE LINK sanjose
DELETE DEVICE ourctc
```

This example shows a DELETE PORT statement that deletes a reservation for port 5001:

```
PORT 5001 TCP MEGA
DELETE PORT 5001 TCP MEGA
```

This example shows a PORT statement that denies all jobs access to unreserved UDP ports on explicit binds. The example also shows the DELETE PORT statement that deletes this access restriction.

```
PORT UNRSV UDP * DENY WHENBIND
DELETE PORT UNRSV UDP *
```

The keywords DENY and WHENBIND are not required on the DELETE PORT statement.

This example shows several PORTRANGE statements to reserve ports for MEGA, and then several DELETE PORTRANGE statements to delete the reservations for those ports:

```
PORTRANGE 5000 10 UDP MEGA
          5100 10 TCP MEGA NOAUTOLOG
          5200 10 UDP MEGA DELAYACKS
          5300 10 TCP MEGA
          5400 10 UDP MEGA
          5500 10 TCP MEGA NOAUTOLOG DELAYACKS
```

```
DELETE PORTRANGE
         5000 10 UDP MEGA
         5100 10 TCP MEGA NOAUTOLOG
         5200 10 UDP MEGA DELAYACKS
         5300 10 TCP MEGA
         5400 10 UDP MEGA
         5500 10 TCP MEGA NOAUTOLOG DELAYACKS
```

Usage notes

The *link_name* of a deleted link remains associated with its device. It cannot be reassigned to a new device while TCP/IP is active.

Summary of DEVICE and LINK statements

Restriction: The DEVICE and LINK statements apply to IPv4 only.

To define an IPv6 interface, you must use the INTERFACE statement. You can also use the INTERFACE statement to define an IPv4 interface for OSA-Express QDIO Ethernet, HiperSockets, and static VIPA. See “Summary of INTERFACE statements” on page 141 for more information.

Overview of DEVICE and LINK statements

z/OS Communications Server allows a single TCP/IP address space to drive multiple instances of any supported device. To configure your devices, add the appropriate DEVICE and LINK statements to the configuration data set. The LINK statements show how to define a network interface link associated with the device and are included with the DEVICE statement for that device type.

Requirements: The following list shows the minimum required statements to define a network interface for use by TCP/IP:

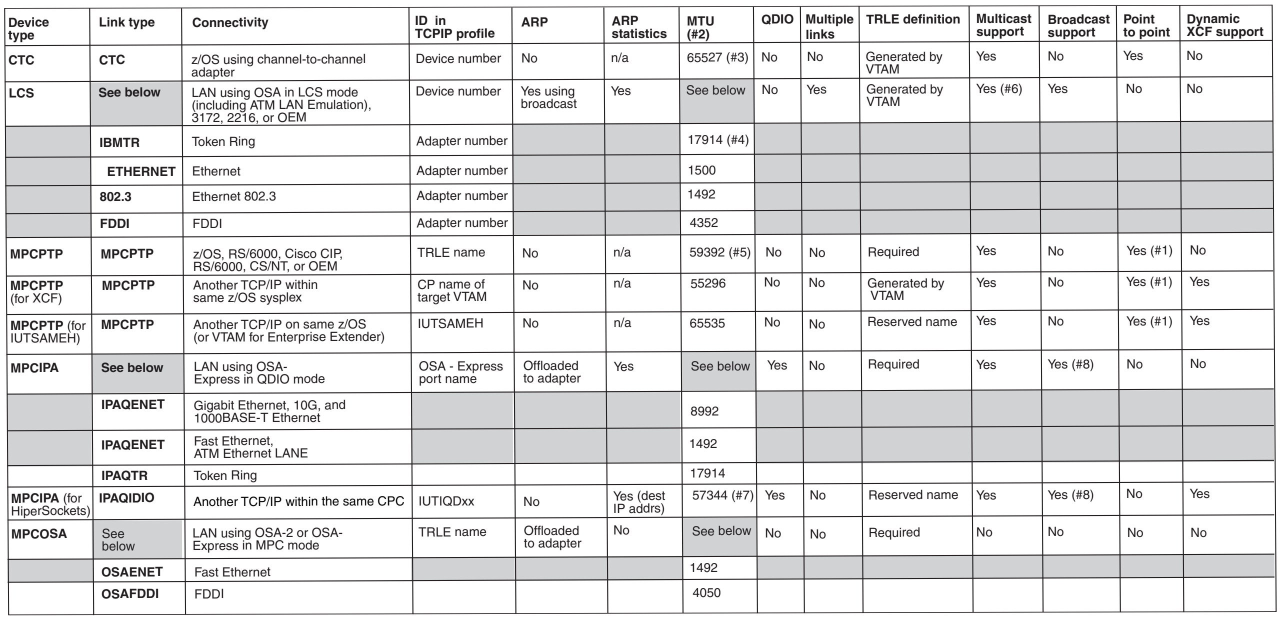
- A set of DEVICE and LINK statements for the appropriate device. Depending on the type of device being defined, additional PROFILE statements, VTAM definitions, or both might be required. For more details, see the DEVICE and LINK statements for the device type.
- A HOME statement assigning an IP address to the LINK interface. For more details, see “HOME statement” on page 136.
- If you are using static routing, define a BEGINROUTES or GATEWAY statement referencing the LINK interface to reach the target networks. For more details, see “BEGINROUTES statement” on page 28 or “GATEWAY statement” on page 109.
- If you are using dynamic routing, see Chapter 11, “OMPROUTE,” on page 483.

Because devices (except VIPA devices) are not automatically initialized, you must also specify a START statement in the configuration data set to start each device automatically.

Restrictions:

- Because TCP/IP has a maximum of 255 started devices (not including VIPA), you cannot start more than 255 devices.
- If you are using OMPROUTE, the maximum number of non-VIPA links that can be specified in the HOME list is 255.
- There is no maximum for static VIPA interfaces, but the maximum number of dynamic VIPA interfaces is 1024.

Figure 1 summarizes information about the various IPv4 network interfaces supported by TCP/IP. The values listed in the MTU column represent the largest MTU supported by each interface.



Device type	Link type	Connectivity	ID in TCPIP profile	ARP	ARP statistics	MTU (#2)	QDIO	Multiple links	TRLE definition	Multicast support	Broadcast support	Point to point	Dynamic XCF support
CTC	**CTC**	z/OS using channel-to-channel adapter	Device number	No	n/a	65527 (#3)	No	No	Generated by VTAM	Yes	No	Yes	No
LCS	**See below**	LAN using OSA in LCS mode (including ATM LAN Emulation), 3172, 2216, or OEM	Device number	Yes using broadcast	Yes	See below	No	Yes	Generated by VTAM	Yes (#6)	Yes	No	No
	IBMTR	Token Ring	Adapter number			17914 (#4)							
	ETHERNET	Ethernet	Adapter number			1500							
	802.3	Ethernet 802.3	Adapter number			1492							
	FDDI	FDDI	Adapter number			4352							
MPCPTP	**MPCPTP**	z/OS, RS/6000, Cisco CIP, RS/6000, CS/NT, or OEM	TRLE name	No	n/a	59392 (#5)	No	No	Required	Yes	No	Yes (#1)	No
MPCPTP (for XCF)	**MPCPTP**	Another TCP/IP within same z/OS sysplex	CP name of target VTAM	No	n/a	55296	No	No	Generated by VTAM	Yes	No	Yes (#1)	Yes
MPCPTP (for IUTSAMEH)	**MPCPTP**	Another TCP/IP on same z/OS (or VTAM for Enterprise Extender)	IUTSAMEH	No	n/a	65535	No	No	Reserved name	Yes	No	Yes (#1)	Yes
MPCIPA	**See below**	LAN using OSA-Express in QDIO mode	OSA - Express port name	Offloaded to adapter	Yes	See below	Yes	No	Required	Yes	Yes (#8)	No	No
	IPAQENET	Gigabit Ethernet, 10G, and 1000BASE-T Ethernet				8992							
	IPAQENET	Fast Ethernet, ATM Ethernet LANE				1492							
	IPAQTR	Token Ring				17914							
MPCIPA (for HiperSockets)	**IPAQIDIO**	Another TCP/IP within the same CPC	IUTIQDxx	No	Yes (dest IP addrs)	57344 (#7)	Yes	No	Reserved name	Yes	Yes (#8)	No	Yes
MPCOSA	See below	LAN using OSA-2 or OSA-Express in MPC mode	TRLE name	Offloaded to adapter	No	See below	No	No	Required	No	No	No	No
	OSAENET	Fast Ethernet				1492							
	OSAFDDI	FDDI				4050							

Figure 1. Summary of DEVICE and LINK statements

Notes:

1. Can be point-to-multipoint.
2. The MTU column represents the largest MTU supported by the interface.
3. Based on the IOBUFFERSIZE value on the CTC device statement in TCP/IP profile.
4. For non-packed mode, the maximum MTU is equal to the write buffer size value on the CLAW device statement in the TCP/IP profile. For packed mode, the maximum MTU is 4096.
5. This MTU value assumes 16Mb Token-Ring. (For 4Mb TR, you must configure an MTU of 4464 or lower.)
6. Based on the read and write buffer size values on the CDLC device statement in TCPIP profile.
7. Based on MAXBFRU value in the TRLE definition.
8. Some LCS devices (for example, old 3172s) do not support multicast.
9. Based on frame size configured in HCD.
10. Requires IPBCAST parameter on LINK statement in TCP/IP profile.

Recovering from device failures

TCP/IP automatically attempts reactivation of the non-VIPA device following some device-failure indications (regardless of the AUTORESTART setting). Specifying AUTORESTART causes TCP/IP to attempt reactivation following most device-failure indications.

The AUTORESTART option is meaningful only for errors that occur after the device is active. For errors that occur before the device reaches the active state, AUTORESTART has no effect, as such errors might likely be the result of a configuration error (for example, incorrect device number specification within the

TCP/IP PROFILE). No automatic error correction would be possible for such an error, and for this reason, TCP/IP initiates device recovery only when evidence of a previously working configuration exists. For any error encountered before the device reaches the active state, the user should correct any configuration error and initiate a new START DEVICE.

If automatic reactivation is attempted, the number of allowable reactivation attempts is determined from the IPCONFIG DEVRETRYDURATION setting.

DEVRETRYDURATION specifies the duration of the Retry Period, during which TCP/IP attempts automatic recovery of a device. The first reactivation attempt is performed two seconds after the original error, and subsequent attempts are 30 seconds apart. If not successfully reactivated within the specified retry duration, the device is returned to the INACTIVE state, and a manual START of the device is required after the error has been corrected.

Missing interrupt handler factors

When multiple subchannels are used for channel-layer communications, WRITE operations and READ operations are separated onto their own subchannels. On a multi-subchannel device, the missing interrupt handler (MIH) is automatically (by VTAM) configured OFF on the READ subchannels. (This is necessary, as a READ command is always active for such devices, and MIH would detect a missing interrupt on the READ subchannels any time the device experienced an idle period.) Therefore, there is no need to specify any MIH values for TCP/IP read devices.

You should configure a reasonable MIH value for the WRITE subchannels on a multi-subchannel device, as well as for the single subchannel on other devices (for example, CDLC), as this protects the system from a storage-usage spike, brought on by a hung device.

VTAM honors the MIH provided for ATM, MPCPTP, MPCPTP6, and MPCOSA write devices, but imposes a limit of four minutes and fifteen seconds. If a value is not provided, VTAM uses a value of 30 seconds, unless running as a guest on VM in which case, a value of 45 seconds is used. For other device types, an MIH value of 0, which disables MIH, for a TCP/IP write device or a single-subchannel device is *not* advisable.

Reasonable values for MIH on the WRITE (or only) subchannel in the range 15-30 seconds [a value of 30 seconds might be warranted if either channel extenders are in the configuration, or dispatching delays (due to running second level, under VM) are possible]. For nonextended ESCON channels, being driven by z/OS running native, 15 seconds is the preferred MIH value.

In summary, MIH on the WRITE (or only) subchannel should be configured ON, with a value in the range 15-30 seconds for the following TCP/IP device types:

- For multi-subchannel TCP/IP device types, set MIH only on the WRITE subchannels:
  - For ATM, MPCPTP, and MPCOSA, the write subchannels are specified on the WRITE parameter of the TRLE definition.
  - For MPCIPA, the WRITE-control subchannel is specified on the WRITE parameter of the TRLE definition.

  - For LCS, CLAW, and Hyperchannel, the WRITE subchannel is *device_number* +1 (where *device_number* is the value specified on the DEVICE statement in the TCP/IP profile).
- For single-subchannel TCP/IP device types, set MIH on the single subchannel.
  - For CDLC, the WRITE subchannel is the *device_number* value specified on the DEVICE statement in the TCP/IP profile.

Tip: To override the default MIH value for a given subchannel, use the MIH statement in the IECIOSxx parmlib member or use the SETIOS MIH command. See z/OS MVS System Commands and z/OS MVS Initialization and Tuning Guide for more information about the IECIOSxx parmlib and SETIOS command, respectively.

Restriction: For all other TCP/IP device types (including the XCF and IUTSAMEH types of MPCPTP and the data devices for MPCIPA), MIH is either not applicable or is automatically disabled by VTAM.

DEVICE and LINK statements relationship to VTAM configuration

z/OS Communications Server provides a set of High Performance Data Transfer (HPDT) services that includes MultiPath Channel (MPC), a high-speed channel interface designed for network protocol use (for example, APPN or TCP/IP). Multiple protocols can either share or have exclusive use of a set of channel paths to an attached platform. The term *MPC+* is used to distinguish this multi-protocol version of MPC from earlier versions that were restricted to APPN usage only.

MPC provides the user with the ability to have multiple device paths defined as a single logical connection. The term *MPC group* is used to define a single MPC connection that can contain multiple read and write paths. The number of read and write paths do not have to be equal, but there must be at least one read and write path defined within each MPC group.

MPC groups are defined using the Transport Resource List (TRL), where each defined MPC group becomes an entry (that is, a TRLE) in the TRL table. The user defines the channel paths that are a part of the group in the TRLE. Each TRLE is identified by a *resource_name*. For ATM, the TRLE also has a *port_name* to identify a particular ATM port. For details about defining a TRLE, see the z/OS Communications Server: SNA Resource Definition Reference.

Modifying DEVICE and LINK statements

To modify most LINK statement parameters (and any DEVICE statement parameters), you must first delete and then redefine the LINK or DEVICE statement.

However, the following LINK statement parameters are dynamically modifiable:
- MONSYSPLEX
- NOMONSYSPLEX

To modify these parameters on a LINK statement, use a VARY TCPIP,,OBEYFILE command with a data set that contains a LINK statement for an existing link name which has new values for these parameters.

Guidelines:
- Any changes to non-modifiable parameters are ignored.

- If any modifiable parameters are not specified, prior values remain in effect for these parameters.

Steps for modifying LINK statements

This topic describe the steps for modifying the LINK statement.

Procedure

Perform the following steps to modify all other parameters on a LINK statement or to modify any DEVICE statement parameters:

1. Stop the device.

   ---

2. Use a VARY TCPIP,,OBEYFILE command with a data set that contains:
   - A new HOME statement that does not contain the home IP address or addresses of the LINK or LINKs involved in the DELETE
   - DELETE linkname and DELETE devicename statements

   ---

3. Use a VARY TCPIP,,OBEYFILE command with a data set that contains:
   - The changed DEVICE and LINK statements
   - A new HOME statement that includes the home IP address or addresses of the LINK or LINKs being added

   The data set used on the VARY TCPIP,,OBEYFILE command in this step should be different from the data set used in step . Do not put the DELETE and redefinition of an interface in the same OBEYFILE data set.

   ---

4. Start the device.

Results

Guideline: To change parameters on a LINK statement only, you do not need to delete the DEVICE name statement and later redefine the DEVICE name statement.

To dynamically change a value on a LINK statement only, do not perform the DELETE devicename and redefine DEVICE steps in "Steps for modifying LINK statements."

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands.

You can add new DEVICE and LINK statements using the VARY TCPIP command. You can also delete and redefine existing statements.

When you add new LINK statements, any corresponding BEGINROUTES, GATEWAY, HOME, and TRANSLATE statements coded to include the new links are treated as replacements for active statements. Therefore, when you code the BEGINROUTES, GATEWAY, HOME, or TRANSLATE statements of the data set specified on a VARY TCPIP,,OBEYFILE command, be sure to include new and existing links that you want to have active in your configuration.

Monitoring network links (DEVICE and LINK statements)

To delete links, the devices must be stopped. When the devices are stopped, the link becomes inactive. If the TCP/IP stack is currently monitoring interfaces and detects that all monitored interfaces are inactive as a result of the devices being stopped, the TCP/IP stack might issue messages about the problem and might trigger a recovery action. You can disable monitoring of these interfaces. To do this, specify the NOMONSYSPLEX keyword on the LINK statement using the VARY TCPIP,,OBEYFILE command before stopping the devices. For more information, see sysplex problem detection and recovery in z/OS Communications Server: IP Configuration Guide.

DEVICE and LINK — ATM devices statement

This device type will not be supported in a future release. It is recommended that you migrate to a later interface type, such as OSA-Express QDIO or HiperSockets.

Use the DEVICE statement to specify the name of the ATM device that you use. Use the LINK statement to define a network interface link associated with the ATM device.

The presence of DEVICE and LINK ATM statements in your *PROFILE.TCPIP* enables ATM native mode and SNMP network management support for the ATM device. Even if an ATM device is not being used by this TCP/IP, or is being used by TCP/IP in ATM LAN Emulation mode instead of Native mode, specifying DEVICE and LINK statements enable you to retrieve SNMP network management data for the ATM device. Enabling SNMP network management data for the ATM devices also requires specification of the OSAENABLED parameter on the SACONFIG Profile statement. For more information about SNMP OSA Management, see z/OS Communications Server: IP Configuration Guide.

You can specify multiple LINKs for an ATM device. This is so an ATM device can be in more than one LIS.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see "Missing interrupt handler factors" on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
                                                  ┌─ENABLEINcomingsvc──┐
►►──DEVice──device_name──ATM──PORTNAME──port_name──┼────────────────────┼──────────►
                                                  └─DISABLEINcomingsvc─┘

   ┌─NOAUTORestart─┐
►──┼───────────────┼─────────────────────────────────────────────────────────────►◄
   └─AUTORestart───┘
```

Parameters

device_name
: The name of the device.

Requirement: The device name must be the Open Systems Adapter (OSA) name known to MPC and OSA/SF. The maximum length is eight characters.

This name, the OSA name, must match the name specified on the transport resource list element (TRLE).For more information about the TRLE, see z/OS Communications Server: SNA Resource Definition Reference. The same name is specified in the LINK statements.

ATM
Specifies the device is for ATM use.

PORTNAME *port_name*
The OSA port name. The maximum length is eight characters.

Requirements:

- This name must match the port name specified on the transport resource list element (TRLE). For more information about the TRLE, see z/OS Communications Server: SNA Resource Definition Reference.
- The PORTNAME must be the same in all instances of TCP/IP and VTAM that share the same adapter.

DISABLEINCOMINGSVC
Device cannot be used for incoming SVCs.

ENABLEINCOMINGSVC
Allow incoming SVC calls for this device; the device can be used for both outgoing and incoming SVCs.

AUTORESTART | NOAUTORESTART
Controls device failure reactivation behavior.

AUTORESTART
In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see “Recovering from device failures” on page 48.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

Syntax

Rule: The optional parameters on the LINK statement following *device_name* can be specified in any order.

```
►►──LINK──link_name──ATM──device_name──┬─────────────────┬──────────────────────►
                                       └─LIS── lis_name──┘


  ┌─IFSPEED 0──────────────┐   ┌─SECCLASS 255────────────┐  ┌─NOMONSYSPLEX─┐
►─┼────────────────────────┼───┼─────────────────────────┼──┼──────────────┼────────►◄
  ├─IFSPEED── ifspeed──────┤   └─SECCLASS security_class─┘  └─MONSYSPLEX───┘
  └─IFHSPEED── ifhspeed────┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters.

ATM
Specifies that the link is an ATM link.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

LIS *lis_name*
The logical IP subnet for this LINK. This parameter is required only if the link is to be used for SVC connections. The maximum length is 16 characters. The *lis_name* must be defined on an ATMLIS statement prior to being used on the LINK statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for the *ifspeed* variable is 0, the maximum value is 2 147 483 647, and the default is 0 set dynamically. Until the interface is successfully started, this value is used by SNMP as the value of the ifSpeed MIB object. After the interface is successfully started, SNMP uses the actual speed reported by the interface as the value of the ifSpeed MIB object. The value of this parameter has no effect on the operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for the *ifhspeed* variable is 0, the maximum value is 2 147, and the default *ifhspeed* for an ATM link is 0. Until the interface is successfully started, this value is used by SNMP as the value of the ifHighSpeed MIB object. After the interface is successfully started, SNMP uses the actual speed reported by the interface as the value of the ifHighSpeed MIB object. The value of this parameter has no effect on the operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

Examples

The following example specifies that OSA1 is an ATM device:

```
DEVICE OSA1  ATM  PORTNAME  PORT1
LINK   LINK1 ATM  OSA1
```

Usage notes

To see samples of commands for using dynamic routing with this device, see the information about NBMA subnetworks in z/OS Communications Server: IP Configuration Guide.

Related topics

- “ATMARPSV statement” on page 17
- “ATMLIS statement” on page 19
- “ATMPVC statement” on page 22
- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- “DELETE statement” on page 43
- “GATEWAY statement” on page 109
- “HOME statement” on page 136
- “SACONFIG statement” on page 271
- “START statement” on page 292
- “STOP statement” on page 293
- “TRANSLATE statement” on page 301

DEVICE and LINK — CLAW devices statement

| This device type will not be supported in a future release. It is recommended that
| you migrate to a later interface type, such as OSA-Express QDIO or HiperSockets.

Use the DEVICE statement to specify the name and hexadecimal device number of a Common Link Access to Workstation (CLAW) device that you use. Devices that use the CLAW protocol include SP2, IBM pSeries servers, and Cisco 7200/7500-series channel-attached routers. Only one DEVICE statement should be used for each device. Use the LINK statement to define a network interface link associated with CLAW devices.

Restriction: Only one LINK statement should be used for each device.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see “Missing interrupt handler factors” on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──DEVice──device_name──CLAW──device_number─────────────────────────────►


                                                    ┌─15───────────┐
►─host_claw_name──workstation_claw_name──┬─NONE───┬─┼──────────────┼─────────►
                                         └─PACKED─┘ └─read_buffers─┘

  ┌─15────────────┐ ┌─4096(Unpacked) or 32K (Packed)─┐
►─┼───────────────┼─┼────────────────────────────────┼──────────────────────►
  └─write_buffers─┘ └─read_size──────────────────────┘

  ┌─4096(Unpacked) or 32K (Packed)─┐ ┌─NOAUTORestart─┐
►─┼────────────────────────────────┼─┼───────────────┼─────────────────────►◄
  └─write_size─────────────────────┘ └─AUTORestart───┘
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CLAW
: Specifies the device is a CLAW device.

device number
: The hexadecimal device number of the pSeries. TCP/IP also uses device number + 1.

host_claw_name
: A value that defines the name of the host system in the system validation exchange between the TCP/IP code and the workstation code. This name must match the HOSTNAME configured on the device.

  The maximum length is eight characters.

workstation_claw_name
: A value for the name of the workstation for the system validation exchange. This name must match the Workstation (or Device) Name configured on the device. The maximum length is eight characters.

NONE
: This CLAW device operates in non-packed mode. This is the default value.

PACKED
: This CLAW device operates in packed mode.

read_buffers
: This is the decimal number (one or more) of buffers to allocate to the read channel program. The minimum value that can be specified for *read_buffers* is 1; the maximum effective value is limited to 256K/Read_Size, even if a larger value is coded on this statement. This should be large enough to give TCP/IP sufficient time to process the received data and append the buffer to the running channel program before it terminates. Each of these buffers uses real storage, so the number should be small enough not to impact overall system performance. The default is 15.

`write_buffers`
This is the decimal number (one or more) of buffers to allocate to the write channel program. The minimum value that can be specified for *write_buffers* is 1; the maximum effective value is limited to 256K/Write_Size, even if a larger value is coded on this statement. This should be large enough that a busy TCP/IP can reuse buffers without the channel program terminating. Each of these buffers uses real storage, so the number should be small enough not to impact overall system performance. The default is 15.

`read_size`
This is the size of the read buffers. If non-packed mode is specified, values are:

- 1024
- 2048
- 3072
- 4096

If packed mode is specified, the valid values for the read_size parameter are:

- 32K
- 60K

The default for non-packed mode is 4096. The default for packed mode is 32K.

Use the following guidelines for selection *read_size*:

Unpacked mode
When configuring CLAW to communicate with pSeries, choose the *read_size* value that matches the transmit buffer size configured on the channel adapter (this is usually 4096, unless the administrator has overridden this setting on the adapter). When configuring CLAW to communicate with a Cisco 7200-series or 7500-series router in non-packed mode, always specify a *read_size* of 4096. For other CLAW devices, see the documentation for the device.

Packed mode
When running workloads that involve bulk-data transfer inbound, the 60K read_size value delivers a higher throughput than the 32K value. However, this larger buffer consumes more REAL storage than the 32K setting.

`write_size`
This is the size of the write buffers. If non-packed mode is specified, values are:

- 1024
- 2048
- 3072
- 4096 or 4K

If packed mode is specified, the valid values for the *write_size* parameter are:

- 32K
- 60K

The default for UnPacked mode is 4096. The default for Packed mode is 32K.

Use the following guidelines for selection *write_size*:

Unpacked mode
When configuring CLAW to communicate with pSeries, choose the *write_size* value that matches the receive buffer size configured on the

channel adapter (this is usually 4096, unless the administrator has overridden this setting on the adapter). When configuring CLAW to communicate with a Cisco 7200-series or 7500-series router in non-packed mode, always specify a *read_size* of 4096. For other CLAW devices, see the documentation for the device.

Packed mode
When running workloads that involve bulk-data transfer outbound, the 60K *write_size* value delivers a higher throughput than the 32K value; however, the larger buffer consumes more REAL storage than the 32K setting.

AUTORESTART | NOAUTORESTART
Controls device failure reactivation behavior.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see "Recovering from device failures" on page 48.

Syntax

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

```
                                                         ┌─IFSPEED 100000000──┐
►►──LINK──link_name──IP──0──device_name──┬──────┬────────┼────────────────────┼──────────►
                                         └─P2MP─┘        ├─IFSPEED── ifspeed──┤
                                                         └─IFHSPEED── ifhspeed┘

   ┌─SECCLASS 255────────────┐  ┌─NOMONSYSPLEX─┐
►──┼─────────────────────────┼──┼──────────────┼──────────────────────────────────────►◄
   └─SECCLASS security_class─┘  └─MONSYSPLEX───┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters.

IP 0
Specifies that the link is an IP link.

device_name
The *device_name* must be the same as specified in the DEVICE statement. The maximum length is 16 characters.

P2MP
Treat this CLAW link as a point-to-multipoint link. The default is point-to-point. Point-to-multipoint RIP neighbors with which OMPROUTE exchanges routing information are learned through RIP_INTERFACE NEIGHBOR statements or upon receipt of an RIP update from the same-subnet neighbor.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* is 0; the maximum value is

2147483647. The default is 100000000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* is 0; the maximum value is 2147. The default is 100. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

Examples

This example shows how you might code DEVICE, LINK, and related statements for a pSeries connection.

```
DEVICE RS6K CLAW 6B2 HOST PSCA NONE
LINK IPLINK1 IP 0 RS6K
HOME
   192.10.10.1 IPLINK1

GATEWAY
;
; Network  First hop   Driver   Packet size  Subnet mask  Subnet value

  192.10.10.2  =         IPLINK1 DEFAULTSIZE  HOST
  DEFAULTNET 192.10.10.2 IPLINK1 DEFAULTSIZE  0

; BSDROUTINGPARMS: Defines the characteristics of each link defined at the host.
;
 ; If not supplied, defaults will be supplied from:
; (1) Static routing definitions in BEGINROUTES
; (2) OMPROUTE configuration (if OMPROUTE is running)
; (3) Stack's interface layer based on hardware capabilites ; and characteristics
: of devices and links.
; - OMPROUTE does not require BSDROUTINGPARMS. However,
; it will override the parameters with its coded or
; defaulted values from its configuration.
; - NCPROUTE requires BSDROUTINGPARMS to route Transport
; PDUs prior to OMPROUTE activation. If OMPROUTE is
; also used, the parameters must match the corresponding
; ones in OMPROUTE configuration for the channel-
; attached links.
;
```

Usage notes

- Claw packing was originally developed to communicate with the Cisco 7200 series routers with Channel Port Adapters (ECPAs or PCPAs) and the Cisco 7500 series routers with Channel Interface Processors (CIPs), but newer router models from Cisco or other vendors might have incorporated the Claw packing function since that time. Please consult your router vendor if there are questions about the packing capability of your router.

  The prerequisite microcode from Cisco is cip26-17 or xcpa26-17 for 12.0 IOS releases and cip27-11 or xcpa27-11 for 12.1 IOS releases; also, any future image that has the following problems resolved: CSCds19174 and CSCds24793.
- If PACKED operation is specified, z/OS Communications Server ensures READ and WRITE buffer sizes of at least 32K, and enforces an interface MTU of 4096 bytes on the z/OS side of the channel. The interface MTU is not to be confused with the MTU value that is defined in the routing definitions; the interface MTU sets an upper limit on what the MTU can be for a routing definition.
- If the z/OS server running the CLAW device driver is a second-level (Virtual not equal Real) guest on a VM system, certain elements of the CLAW protocol are transparently disabled. In particular, the effects of extending the channel program are seen, and this can result in a higher interrupt rate with potentially lower throughput. Other than this slightly degraded performance, the CLAW device driver is functional in a Virtual-Not-Equal-Real guest.

Related topics

- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- “GATEWAY statement” on page 109
- “HOME statement” on page 136
- “START statement” on page 292
- “STOP statement” on page 293

DEVICE and LINK — CTC devices statement

Use the DEVICE statement to specify the name and hexadecimal device number of the channel-to-channel (CTC) devices that you use. Use the LINK statement to define a network interface link associated with the CTC devices.

Requirement: You must use a separate DEVICE statement for each device you use. The same is true for the LINK statement.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see “Missing interrupt handler factors” on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──DEVice──device_name──CTC──base_device_number──────────────────────────►

   ┌─IOBUFFERSIZE 32768──────────┐ ┌─NOAUTORestart─┐
►──┼─────────────────────────────┼─┼───────────────┼──────────────────────►◄
   └─IOBUFFERSIZE── buffer_size──┘ └─AUTORestart───┘
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CTC
: Specifies the device is a channel-to-channel (CTC) device.

base_device_number
: The hexadecimal base device number associated with the CTC adapter. Two numbers are used by TCP/IP: the *base_device_number* and *base_device_number+1*.

IOBUFFERSIZE *buffer_size*
: Specifies the I/O buffer size. The buffer size must be 32K (minimum), 32 768 (default), or 65 535 (maximum).

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

NOAUTORESTART
: For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
: In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see “Recovering from device failures” on page 48.

Syntax

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

```
                                                ┌─IFSPEED 4500000────┐
►►──LINK──link_name──CTC──adapter_addr──device_name──┼────────────────────┼──►
                                                ├─IFSPEED── ifspeed──┤
                                                └─IFHSPEED── ifhspeed┘
```

```
   ┌─SECCLASS 255────────────┐  ┌─NOMONSYSPLEX─┐
►──┼─────────────────────────┼──┼──────────────┼──────────────────────────►◄
   └─SECCLASS security_class─┘  └─MONSYSPLEX───┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters.

CTC
Specifies that the link is a channel-to-channel link.

adapter_addr
An integer used to specify whether the DEVICE statement's parameter, *base_device_number*, is the read device number or the write device number. Use 0 to indicate that the base device number is the read device and 1 to indicate that the *base_device_number* is the write device.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* for a CTC link is 0; the maximum value is 2 147 483 647. The default is 4 500 000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* for a CTC link is 0; the maximum value is 2147. The default is 4. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See "Modifying DEVICE and LINK statements" on page 50 for modifying information.

Usage notes

The configured I/O buffer sizes at each end of the CTC connection must match. A buffer size mismatch can cause packet loss or I/O errors, resulting in deactivation of the CTC connection. CTC I/O buffer size can be explicitly specified with the IOBUFFERSIZE parameter.

Related topics

- "BEGINROUTES statement" on page 28
- "BSDROUTINGPARMS statement" on page 36
- "GATEWAY statement" on page 109
- "HOME statement" on page 136
- "START statement" on page 292
- "STOP statement" on page 293

DEVICE and LINK — HYPERchannel A220 devices statement

| This device type will not be supported in a future release. It is recommended that
| you migrate to a later interface type, such as OSA-Express QDIO or HiperSockets.

Use the DEVICE statement to specify the name and hexadecimal device number of the HYPERchannel A220 device.

Use the LINK statement to define the link to the HYPERchannel A220 adapter.

The TRANSLATE statement is required for HYPERchannel A220 devices.

Some token-ring hardware does not recognize the RFC 1469 mandated functional MAC address for multicast. The TRANSLATE statement can be used to configure a token-ring link to broadcast multicast datagrams as an alternative to using the functional MAC address. Use the reserved class D address 224.0.0.0 with one of the following special physical addresses:

- FFFFFFFFFFFF for all rings broadcast
- C00000040000 to reset back to the default functional address

The following examples show how to specify each method:

- All rings:

```
TRANSLATE
  224.0.0.0  IBMTR  FFFFFFFFFFFF  linkname
```

- Assigned functional address:

```
TRANSLATE
  224.0.0.0  IBMTR  C00000040000  linkname
```

The TRANSLATE statement is effective on a per link basis. You do not have to code a TRANSLATE statement if you want the assigned functional address, as it is the default method.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see “Missing interrupt handler factors” on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
                                            ┌─NOAUTORestart─┐
►►──DEVICE──device_name──HCH──base_device_number──┼───────────────┼──────────────►◄
                                            └─AUTORestart───┘
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

HCH
: Specifies the device is a HYPERchannel A220.

base_device_number
: The hexadecimal base device number (in the range 0 - FFFF) associated with the A220 adapter. Two addresses are used by TCP/IP: the *base_device_number* and *base_device_number+1*.

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

NOAUTORESTART
: For most device failures, specifying the NOAUTORESTART value indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
: In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see “Recovering from device failures” on page 48.

Syntax

Rule: The optional parameters on the LINK statement following *device_name* can be specified in any order.

```
                                               ┌─IFSPEED 50000000───┐
►►──LINK──link_name──HCH──adapter_addr──device_name──┼────────────────────┼──────────►
                                               ├─IFSPEED── ifspeed──┤
                                               └─IFHSPEED── ifhspeed┘
```

```
   ┌─SECCLASS 255────────────┐   ┌─NOMONSYSPLEX─┐
►──┼─────────────────────────┼───┼──────────────┼──────────────────────────►◄
   └─SECCLASS security_class─┘   └─MONSYSPLEX───┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters.

HCH
Specifies that the link is a HYPERchannel A220.

adapter_addr
This value must be an integer, but the value is ignored. This parameter is included for consistency with LINK statement formats for other device types.

device_name
The *device_name* must be the same as specified in the DEVICE statement. The maximum length is 16 characters.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* for a hyperchannel link is 0; the maximum value is 2 147 483 647. The default is 50 000 000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* for a hyperchannel link is 0; the maximum value is 2147. The default is 50. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG

SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See "Modifying DEVICE and LINK statements" on page 50 for modifying information.

Usage notes

- The ATTENTION+BUSY and unit check conditions are normally handled in the background and can affect performance without any visible evidence. The guidelines for HYPERchannel A222 and A223 Mode Switch Settings are:
  - The *Disable Attentions* setting on the HYPERchannel box eliminates the ATTENTION+BUSY status in response to read commands, which reduces overhead.
  - The *Enable Command Retry* setting reduces the number of unit checks needed because of trunk contention. This setting improves performance, because the TCP/IP device driver waits 10 milliseconds before trying a command that produced a unit check again. This setting also eliminates the need to perform sense operations and try commands again.
- To use dynamic routing with this device, see the NBMA subnetworks information (Non_Broadcast parameter) in "OSPF_INTERFACE statement" on page 493 and see examples in z/OS Communications Server: IP Configuration Guide.

Related topics

- "BEGINROUTES statement" on page 28
- "BSDROUTINGPARMS statement" on page 36
- "GATEWAY statement" on page 109
- "HOME statement" on page 136
- "START statement" on page 292
- "STOP statement" on page 293
- "TRANSLATE statement" on page 301

DEVICE and LINK — LAN Channel Station and OSA devices statement

Use the DEVICE statement to specify the name and hexadecimal device number of an IBM 8232 LAN Channel Station (LCS) device, an IBM 3172 Interconnect Controller, an IBM 2216 Multiaccess Connector Model 400, an IBM FDDI, Ethernet, Token-Ring OSA, or an IBM ATM OSA-2 in LAN emulation mode.

Use the LINK statement to define a network interface link associated with an LCS device. The LINK statements used are the Ethernet Network LCS LINK statement, the Token-Ring Network or PC Network LCS LINK statement, and the FDDI LCS LINK statement.

Requirement: You must use a separate LINK statement for each link associated with an LCS device.

Each network interface on the OSA is considered a separate DEVICE. For example, if you are using both ports on the OSA-2 card, you need to code a DEVICE and

LINK pair for each port. For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see “Missing interrupt handler factors” on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
                                             ┌─NONETMAN─┐
►►──DEVice──device_name──LCS──device_number──┤          ├──────────────────────────►
                                             └─NETMAN───┘

  ┌─IOBUFFERSIZE 20480───────┐ ┌─NOAUTORESTART─┐
►─┤                          ├─┤               ├──────────────────────────────────►◄
  └─IOBUFFERSIZE── buffer_size┘ └─AUTORESTART───┘
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified on the LINK statements.

LCS
: Specifies the device is a LAN Channel Station.

device_number
: The hexadecimal device number (in the range 0 - FFFF) of the LCS. *device_number* +1 is also used by the TCP/IP address space.

NETMAN
: Specifies that this device is a 3172 that supports the IBM Enterprise-specific MIB variables for 3172.

  Requirement: NETMAN must be coded before IOBUFFERSIZE.

NONETMAN
: Specifies that this device is not used for NETMAN data retrieval.

IOBUFFERSIZE *buffer_size*
: Specifies the I/O buffer size. The buffer size must be 20K, 20 480, 32K, or 32 768.

  Guidelines:

  - The configured I/O buffer sizes for the host and for the device must match. A buffer size mismatch can cause packet loss or I/O errors, which results in the deactivation of the LCS connection.
  - If the LCS device supports an option to configure a 32K buffer size, then configuring both the device and the TCP/IP profile to 32K provides the best performance. If the device does not support this option, then specify (or default) to 20K in the TCP/IP profile.

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

  NOAUTORESTART
  : For most device failures, specifying the NOAUTORESTART value indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
: In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see “Recovering from device failures” on page 48.

LINK statement for Ethernet network LCS

This LINK statement is used to define an Ethernet link on an IBM 3172 Interconnect Controller and IBM 8232 LAN Channel Station (LCS) or OSA device.

Syntax

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

```
►►──LINK──link_name──┬─ETHERNet─────┬──link_number──device_name─────────────────►
                     ├─802.3────────┤
                     └─ETHEROR802.3─┘

   ┌─IFSPEED 4000000──────┐  ┌─SECCLASS 255─────────────┐  ┌─NOMONSYSPLEX─┐
►──┼──────────────────────┼──┼──────────────────────────┼──┼──────────────┼──────►◄
   ├─IFSPEED  ──ifspeed───┤  └─SECCLASS security_class──┘  └─MONSYSPLEX───┘
   └─IFHSPEED── ifhspeed──┘
```

Parameters

`link_name`
: The name of the link. The maximum length is 16 characters.

`ETHERNET`
: Standard Ethernet protocol only.

`802.3`
: IEEE 802.3 protocol only.

`ETHERor802.3`
: Both standard Ethernet and IEEE 802.3 protocols. When ETHERor802.3 is specified, address resolution packets (ARP) for both protocols are generated. All devices on the network must be able to process or discard these packets.

`link_number`
: The relative adapter number (0 for the first Ethernet protocol network in the LCS, 1 for the second Ethernet protocol network, and so on). If defining OSA, this value is the port number on the OSA.

`device_name`
: The *device_name* must be the same name as specified in the DEVICE statement. The maximum length is 16 characters.

`IFSPEED` `ifspeed`
: An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* for an LCS link is 0, the maximum value is 2 147 483 647. The default is 4 000 000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

`IFHSPEED` `ifhspeed`
: An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* for an

LCS link is 0, the maximum value is 2147. The default is 4. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

LINK statement for token-ring network or PC network LCS

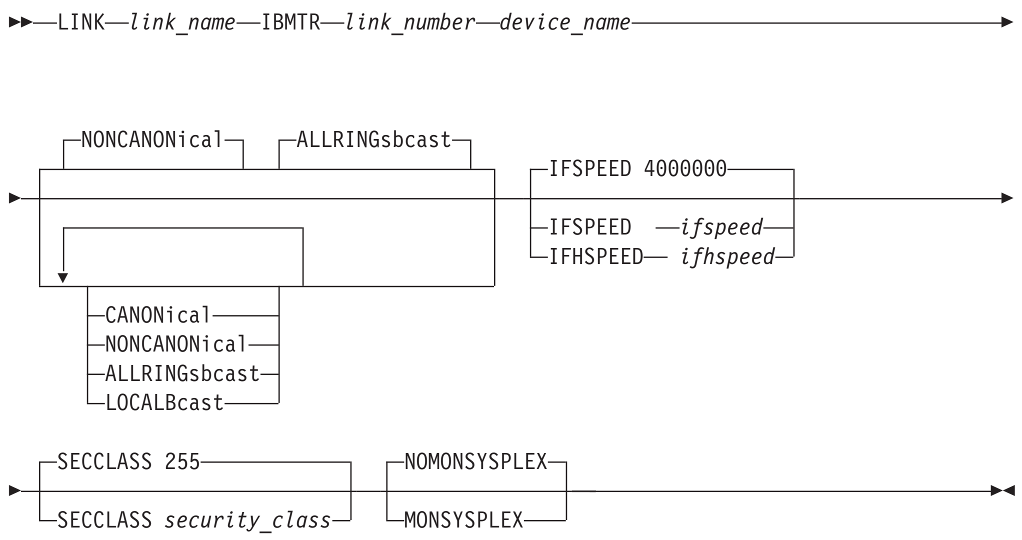
The token-ring LCS LINK statement is used to define the token-ring link to the LCS (IBM 8232 or IBM 3172) or OSA device previously defined by the LCS DEVICE statement. By default, the token-ring LCS LINK statement is also used to define the PC Network link.

Medium Access Control (MAC) addresses in the Address Resolution Protocol (ARP) packets on this token-ring network are in the more common, noncanonical format.

Requirement: All TCP/IP hosts and gateways on a given token-ring network must be configured to use the same form for MAC addresses in ARP packets, either canonical or noncanonical. For more information about the terms, canonical and noncanonical, see IEEE standards 802.3 and 802.5.

Rule: The optional parameters on the LINK statement following *device_name* can be specified in any order.

Syntax



Parameters

link_name
: The name of the link. The maximum length is 16 characters.

IBMTR
: Specifies that the link is to an IBM Token-Ring.

link_number
: The relative adapter number (0 for the first token-ring adapter in the LCS, 1 for the second token-ring, and so on). If defining OSA, this value is the port number on the OSA.

device_name
: The *device_name* must be the same as specified in the DEVICE statement. The maximum length is 16 characters.

CANONICAL
: MAC addresses in Address Resolution Protocol (ARP) packets on this token-ring network are in the canonical IEEE 802.5 form.

NONCANONICAL
: MAC addresses in ARP packets on this token-ring network are in the more common noncanonical format. This is the default value.

ALLRINGSBCAST
: All IP and ARP broadcasts are sent as all-rings broadcasts, which are propagated through token-ring bridges (Source Route Bridging). This is the default value.

LOCALBCAST
: All IP and ARP broadcasts are sent only on the local ring and are not propagated through token-ring bridges (Transparent Bridging).

IFSPEED *ifspeed*
: An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
: An optional estimate of the interface's current bandwidth in one million bits

per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

LINK statement for FDDI LCS

The following topic discusses the LINK statement for FDDI LCS.

This LINK statement is used to define the Fiber Distributed Data Interface (FDDI) link to the LCS (IBM 3172 Models 002 and 003) or OSA device defined by the LCS DEVICE statement.

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

Syntax

```
                                                  ┌─IFSPEED 4000000──────┐
►►──LINK──link_name──FDDI──link_number──device_name─┼──────────────────────┼──────────►
                                                  ├─IFSPEED  ──ifspeed──┤
                                                  └─IFHSPEED── ifhspeed─┘

   ┌─SECCLASS 255────────────┐  ┌─NOMONSYSPLEX─┐
►──┼─────────────────────────┼──┼──────────────┼──────────────────────────────────►◄
   └─SECCLASS security_class─┘  └─MONSYSPLEX───┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters.

FDDI
Specifies that the link is to an FDDI network.

link_number
The relative adapter number (0 for the first FDDI adapter in the LCS, 1 for the second FDDI adapter, and so on). If defining OSA, this value is the port number on the OSA.

device_name
The *device_name* must be the same as specified in the DEVICE statement. The maximum length is 16 characters.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See "Modifying DEVICE and LINK statements" on page 50 for modifying information.

Examples

- In this example, LCS1 is a 3172 model 1 with a Token-Ring and Ethernet adapter.

```
DEVICE LCS1   LCS           BA0
LINK TR1   IBMTR    0 LCS1
LINK ETH1  ETHERNET 1 LCS1
```

- In this example, LCS2 is a 3172 model 2 with an FDDI adapter.

```
DEVICE LCS2   LCS           BE0
LINK FDDI1 FDDI     0 LCS2
```

- This example shows how you might code DEVICE, LINK, and related statements for an LCS connection.

```
DEVICE LCS1 LCS BA0
LINK TR1  IBMTR    0 LCS1
LINK TR2  IBMTR    1 LCS1 LOCALBCAST
LINK ETH1 ETHERNET 0 LCS1
HOME
   192.10.10.10  TR1
   9.67.43.10    TR2
   128.50.17.1   ETH1

   GATEWAY
;
; Network  First hop   Driver   Packet size  Subnet mask  Subnet value

  192.10.10    =         TR1       2000       0
  9            =         TR2       2000       0.255.255.0  0.67.43.0
  128.50       =         ETH1      1500       0.0.240.0    0.0.16.0
  DEFAULTNET 9.67.43.1   TR2    DEFAULTSIZE   0

;
;  link     maxmtu   metric   subnet mask     dest addr
; BSDROUTINGPARMS false
;    TR1       2000       0     255.255.255.0   0
;    TR2       2000       0     255.255.255.0   0
;    ETH1      1500       0     255.255.240.0   0
; ENDBSDROUTINGPARMS
;

START LCS1
```

- In this example of an OSA-2 card, LCS1 is Token-Ring Port 0 and LCS2 is an ETHERNET Port 1.

```
DEVICE LCS1   LCS           BA0
LINK TR1   IBMTR    0 LCS1
DEVICE LCS2   LCS           BA2
LINK ETH1  ETHERNET 1 LCS1
```

Usage notes

When an OSA-Express feature is being shared between multiple stacks (OSA port sharing), you need to consider how to configure the OSA address table (OAT). Use the OAT definitions to control the stack to which OSA sends datagrams and ARP packets for specific destination IP addresses. OSA also allows two TCP/IP stacks sharing the port to act as IP routers: a PRIMARY stack and a SECONDARY stack. When an OAT is configured, OSA processes inbound packets as follows:

- If the IP address is configured for a given stack in the OAT, then OSA sends the packet to that stack.
- Otherwise, if a PRIMARY entry is defined in the OAT, then OSA sends the packet to the stack configured as PRIMARY (if active).
- Otherwise, if a SECONDARY entry is defined in the OAT, then OSA sends the packet to the stack configured as SECONDARY (if active).
- Otherwise, OSA discards the packet.

Guideline: Configure the OAT as follows:
- Always configure an OAT entry containing the TCP/IP HOME address associated with the LINK defined in the TCP/IP Profile.
- If you are using Virtual IP Addressing (VIPA) on the LAN, configure OAT entries containing the TCP/IP stack's Virtual IP Addresses.
- If the stack has multiple OSAs onto the same LAN, then configure an OAT entry for the HOME IP address of each of these OSAs. This enables you to take advantage of the fault tolerance provided by the interface takeover (ARP takeover) function.
- To enable a TCP/IP stack to act as a router, configure one of the OAT entries as PRIMARY, and enable IP forwarding (IPCONFIG DATAGRAMFWD in the TCP/IP Profile) on the TCP/IP stack acting as PRIMARY. Likewise, to enable a second stack to back up the PRIMARY router, configure one of the OAT entries as SECONDARY, and enable IP forwarding on the stack acting as SECONDARY. For more information, see *System z9 and zSeries Open Systems Adapter-Express (OSA-Express) Customer's Guide and Reference*.

Related topics
- "BEGINROUTES statement" on page 28
- "BSDROUTINGPARMS statement" on page 36
- "GATEWAY statement" on page 109
- "HOME statement" on page 136
- "START statement" on page 292
- "STOP statement" on page 293
- "TRANSLATE statement" on page 301

DEVICE and LINK — MPCIPA OSA-Express QDIO devices statement

To define an OSA-Express device in QDIO mode, use the MPCIPA DEVICE statement, specifying the PORTNAME value from the TRLE definition as the *device_name* value.

Requirements:
- The TRLE must be defined as MPCLEVEL=QDIO.
- If multiple LPARs share the same OSA, although the TRLE name must be unique in each VTAM, the PORTNAME value must be the same in each of these TRLE definitions for that QDIO interface.

For details about defining a TRLE, see z/OS Communications Server: SNA Resource Definition Reference.

Use the LINK statement to define a network interface link associated with the QDIO interface.

Restriction: Only one LINK statement can be specified for each MPCIPA device.

Tip: You can also use the INTERFACE statement to define an IPv4 interface for OSA-Express QDIO Ethernet, which combines the definitions of the DEVICE, LINK, and HOME statements into a single statement.

Restriction: For a given OSA-Express feature, a device and link definition is precluded in the following scenarios:
- An IPv4 interface is or was previously defined
- Multiple IPv6 definitions are or were previously defined.

When you start an MPCIPA device, TCP/IP registers all non-loopback (home) IPv4 addresses for this TCP/IP instance to OSA-Express feature. (For a dynamic VIPA that exists on multiple stacks, only the stack that owns the DVIPA, and therefore advertises the DVIPA to routers, registers the DVIPA to OSA-Express feature.) This enables the device to route datagrams destined for those IPv4 addresses to this TCP/IP instance. If a datagram is received at this device for an unknown IP address, the device routes the datagram to the TCP/IP instance, depending on the configuration of a virtual MAC (VMAC) address or definition of an instance as PRIROUTER or SECROUTER. If the datagram is not destined for a virtual MAC address and no active TCP/IP instance using this device is defined as PRIROUTER or SECROUTER, the device discards the datagram. For more details about OSA-Express feature routing considerations, see router information in z/OS Communications Server: IP Configuration Guide and primary and secondary routing in z/OS Communications Server: SNA Network Implementation Guide.

If you subsequently add, delete, or change any home IPv4 addresses on this TCP/IP instance, TCP/IP dynamically registers the changes to OSA-Express.

SNMP management data specific to OSA-Express and OSA-Express2 features is supported by the z/OS Communications Server SNMP TCP/IP subagent, and the OSA-Express Direct SNMP subagent. For more information about these SNMP subagents, see TCP/IP subagent and OSA-Express Direct subagent in the z/OS Communications Server: IP Configuration Guide.

For detailed instructions on setting up an OSA-Express feature, see zEnterprise System and System z10 OSA-Express Customer's Guide and Reference.

To determine the OSA-Express microcode level, use the DISPLAY TRL command. If a specific OSA-Express feature is documented with a minimum microcode level, you can use this command to determine whether that function is supported. IBM service might request the microcode level for problem diagnosis. For more information about the DISPLAY TRL command, see z/OS Communications Server: SNA Operation.

For more information about configuring OSA-specific SNMP support, see Step 4: Configure the Open Systems Adapter (OSA) support in z/OS Communications Server: IP Configuration Guide.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see “Missing interrupt handler factors” on page 49.

The following OSA-Express features can be defined in QDIO mode:
- Fast Ethernet
- ATM LANE Ethernet

- Gigabit Ethernet
- Token Ring
- 1000BASE-T Ethernet
- 10G Ethernet

Syntax

Rule: Specify the parameters in the order shown here.

```
                                 ┌─NONRouter─┐  ┌─NOAUTORestart─┐
►►──DEVice──device_name──MPCIPA──┼───────────┼──┼───────────────┼──────────────►◄
                                 ├─PRIRouter─┤  └─AUTORestart───┘
                                 └─SECRouter─┘
```

Parameters

device_name
: The name of the device. The device name must be the PORT name of the LAN adapter defined in a TRLE for a QDIO connection. The maximum length is eight characters.

MPCIPA
: Specifies the device belongs to the MPC family of interfaces and uses the interface based on IP assist.

NONROUTER
: If a datagram is received at this device for an unknown IP address, the datagram is not routed to this TCP/IP instance. This is the default value.

  PRIRouter and SECRouter parameters interact with the VLANID parameter on the LINK statement. See the VLANID parameter to understand this relationship.

  Rule: This keyword is ignored if the VMAC parameter is configured on the LINK statement.

PRIROUTER
: If a datagram is received at this device for an unknown IP address and is not destined for a virtual MAC address, the datagram is routed to this TCP/IP instance.

  Rule: This keyword is ignored if the VMAC parameter is configured on the LINK statement.

SECROUTER
: If a datagram is received at this device for an unknown IP address and is not destined for a virtual MAC address, and there is no active TCP/IP instance defined as PRIROUTER, then the datagram can be routed to this TCP/IP instance. In this case, OSA routes to only one of the active TCP/IP instances that is defined with SECROUTER parameter. This parameter indicates that the OSA considers this TCP/IP instance to be one of the secondary routers.

  Rule: This keyword is ignored if the VMAC parameter is configured on the LINK statement.

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

NOAUTORESTART
: For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
: In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see "Recovering from device failures" on page 48.

Syntax

Rule: The optional parameters on the LINK statement following *device_name* can be specified in any order.

```
►►──LINK──link_name──IPAQENET──device_name──┬─────────┬──┬───────────────┬──────────►
                                            └─IPBCAST─┘  └─VLANID ──id───┘

   ┌─READSTORAGE GLOBAL──────┐  ┌─INBPERF BALANCED────────────┐
►──┼─────────────────────────┼──┼─────────────────────────────┼──────────────────────►
   └─READSTORAGE──┬─MAX─┬────┘  └─INBPERF──┬─DYNAMIC────┬─────┘
                  ├─AVG─┤                  ├─MINCPU─────┤
                  └─MIN─┘                  └─MINLATENCY─┘

   ┌─IFSPEED 100000000──────┐  ┌─SECCLASS 255────────────┐  ┌─NOMONSYSPLEX─┐
►──┼────────────────────────┼──┼─────────────────────────┼──┼──────────────┼──────────►
   ├─IFSPEED── ifspeed──────┤  └─SECCLASS security_class─┘  └─MONSYSPLEX───┘
   └─IFHSPEED── ifhspeed────┘

   ┌─NODYNVLANREG─┐
►──┼──────────────┼──┬──────────────────────────────────────────┬─────────────────►◄
   └─DYNVLANREG───┘  │                       ┌─ROUTEALL─┐       │
                     └─VMAC──┬─────────┬─────┼──────────┼───────┘
                             └─macaddr─┘     └─ROUTELCL─┘
```

Parameters

link_name
: The name of the link. The maximum length is 16 characters.

IPAQENET
: Indicates that the link uses the interface bases on IP assist, belongs to the QDIO family of interfaces, and uses the Gigabit Ethernet or Fast Ethernet protocol. IPAQGNET is accepted for migration purposes.

device_name
: The *device_name* must be the same as specified in the DEVICE statement.

IPBCAST
: Specifies that the link both sends and receives IP broadcast packets. If this parameter is not specified, no IP broadcast packets are sent or received on this link.

VLANID *id*
: An optional parameter followed by a decimal number indicating the virtual LAN identifier to be assigned to this OSA-Express Link. This field should be a virtual LAN identifier recognized by the switch for the LAN connected to this OSA-Express. The valid range is 1 - 4094.

The VLANID parameter interacts with the PRIROUTER and SECROUTER parameters on the DEVICE statement. If you configure both a VLANID and either the PRIROUTER or SECROUTER parameter, then this TCP/IP instance acts as a router for this VLAN ID only. Datagrams that are received at this device for an unknown IP address and are not destined for a virtual MAC are routed only to this TCP/IP instance if the datagrams are VLAN tagged with a matching VLAN ID. For more information about VLANID parameter interactions, see z/OS Communications Server: IP Configuration Guide.

READSTORAGE

An optional parameter indicating the amount of fixed storage that z/OS Communications Server should keep available for read processing for this adapter. The QDIOSTG VTAM start option allows you to specify a value which applies to all OSA-Express adapters in QDIO mode. You can use the READSTORAGE keyword to override the global QDIOSTG value for this adapter based on the inbound workload you expect over this adapter on this stack. The valid values are:

GLOBAL
The amount of storage is determined by the QDIOSTG VTAM start option. This is the default value.

MAX Use this value if you expect a heavy inbound workload over this adapter.

AVG Use this value if you expect a medium inbound workload over this adapter.

MIN Use this value if you expect a light inbound workload over this adapter.

Tip: See the description of the QDIOSTG VTAM start option in the z/OS Communications Server: SNA Resource Definition Reference for details about exactly how much storage is allocated by z/OS Communications Server for each of these values.

Rule: If you define both a LINK and INTERFACE statement for the same adapter, then the READSTORAGE value on the LINK statement must match the READSTORAGE value on the corresponding INTERFACE statement. If you define a LINK statement that contains a value for READSTORAGE that conflicts with the READSTORAGE value for a previous INTERFACE statement for the same adapter, then TCP/IP rejects the LINK statement.

INBPERF

An optional parameter indicating how frequently the adapter should interrupt the host for inbound traffic.

There are three supported static settings (MINCPU, MINLATENCY, and BALANCED). The static settings use static interrupt-timing values. The static values are not always optimal for all workload types or traffic patterns, and cannot account for changes in traffic patterns.

There is also one supported dynamic setting. This setting causes the host (stack) to dynamically adjust the timer-interrupt value while the device is active and in use. This function exploits an OSA hardware function called dynamic LAN idle. Unlike the static settings, the DYNAMIC setting reacts to changes in traffic patterns, and sets the interrupt-timing values at the point where throughput is maximized.

Valid settings for this setting are:

BALANCED
This setting uses a static interrupt-timing value, selected to achieve reasonably high throughput and reasonably low CPU consumption. This is currently the default value.

DYNAMIC
This setting causes the host to dynamically signal the OSA-Express feature to change the timer-interrupt value, based on current inbound workload conditions. The DYNAMIC setting is effective only for OSA-Express2 or later features on at least an IBM System z9® with the corresponding dynamic LAN idle functional support. See the 2094DEVICE Preventive Service Planning (PSP) and the 2096DEVICE Preventive Service Planning (PSP) buckets for more information about the level of OSA-Express2 adapter that supports this function. See the 2097DEVICE Preventive Service Planning (PSP) bucket for more information about the OSA-Express3 adapter that supports this function. When this setting is specified for an Open Systems Adapter-Express that does not support the dynamic LAN idle function, the stack reverts to using the BALANCED setting. The DYNAMIC setting can decrease latency and provide increases in throughput for many interactive workloads. For all other workload combinations, this setting provides performance similar to the three static settings.

Restriction: The QDIO inbound workload queueing function, which is part of the DYNAMIC setting for OSA-Express3 or later features, is not supported for IPAQENET interfaces that are defined by DEVICE and LINK statements. You must convert your IPAQENET definitions to use the INTERFACE statement in order to enable this function. See the information about QDIO inbound workload queueing and Steps for converting from IPv4 IPAQENET DEVICE, LINK, and HOME definitions to the IPv4 IPAQENET INTERFACE statement in z/OS Communications Server: IP Configuration Guide.

MINCPU
This setting uses a static interrupt-timing value, selected to minimize host interrupts without regard to throughput. This mode of operation might result in minor queueing delays (latency) for packets into the host, which is not optimal for workloads with demanding latency requirements.

MINLATENCY
This setting uses a static interrupt-timing value, selected to minimize latency (delay), by more aggressively presenting received packets to the host. This mode of operation generally results in higher CPU consumption than the other three settings. Use this setting only if host CPU consumption is not an issue.

Rule: If you define both a LINK IPAQENET and an INTERFACE IPAQENET6 statement for the same adapter, then the following rules apply for the INBPERF parameter on these statements:

- The value on the LINK statement must match the INBPERF value on the corresponding INTERFACE statement.
- The INTERFACE statement supports the subparameters WORKLOADQ and NOWORKLOADQ for the INBPERF DYNAMIC parameter. These subparameters are associated with QDIO inbound workload queueing support and are not supported on the LINK IPAQENET statement. So, if you specify the INBPERF DYNAMIC parameter for both the LINK and the INTERFACE statements, then you must use the default or specify the

NOWORKLOADQ subparameter for the INBPERF DYNAMIC parameter on the INTERFACE statement. This ensure that the INBPERF DYNAMIC setting for both statements is the same.

- If you define a LINK IPAQENET statement that contains a value for INBPERF that conflicts with the INBPERF value for a previous INTERFACE IPAQENET6 statement for the same adapter, then TCP/IP rejects the LINK statement.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for the *ifspeed* variable for an MPCIPA link is 0; the maximum value is 2147483647. The default is 100 000 000. Until the interface is successfully started, this value is used by SNMP as the value of the ifSpeed MIB object. After the interface is successfully started, SNMP uses the actual speed reported by the interface as the value of the ifSpeed MIB object. The value of this parameter has no effect on the operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for the *ifhspeed* variable for an MPCIPA link is 0; the maximum value is 2147. The default is 100. Until the interface is successfully started, this value is used by SNMP as the value of the ifHighSpeed MIB object. After the interface is successfully started, SNMP uses the actual speed reported by the interface as the value of the ifHighSpeed MIB object. The value of this parameter has no effect on the operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

DYNVLANREG | NODYNVLANREG

Controls VLAN ID configuration behavior for this link.

Restriction: This parameter is applicable only if a VLAN ID is specified on the statement.

Dynamic registration of VLAN IDs is handled by the OSA-Express feature and the physical switch on your LAN. Therefore, both must be at a level that provides the necessary hardware support for dynamic VLAN ID registration, in order for the DYNVLANREG parameter to be effective. After the link is active, you can view the Netstat DEVLINKS/**-d** report output to determine whether your OSA-Express feature can support VLAN dynamic registration. This Netstat report also displays whether or not dynamic VLAN ID registration has been configured for the link.

Rule: If you define both a LINK and INTERFACE statement for the same adapter, then the dynamic VLAN ID registration parameter value on the LINK statement must match the value of this same parameter on the corresponding INTERFACE statement. If you define a LINK statement that contains a dynamic VLAN ID registration parameter value that conflicts with the same parameter value for a previous INTERFACE statement for the same OSA-Express feature, then TCP/IP rejects the LINK statement.

NODYNVLANREG

Specifies that if a VLAN ID is configured for this link, the ID must be manually registered with the physical switches on the corresponding LAN. This is the default value. If this parameter is specified without a VLAN ID, it is ignored.

DYNVLANREG

Specifies that if a VLAN ID is configured for this link, the ID is dynamically registered with the physical switches on the corresponding LAN. If this parameter is specified without a VLAN ID, message EZZ0056I is issued and the NODYNVLANREG setting is used instead.

VMAC *macaddr*

Indicates the virtual MAC address, which can be represented by 12 hexadecimal characters. The OSA-Express feature uses this address rather than the physical MAC address of the device for all IPv4 packets to and from this TCP/IP stack.

The *macaddr* value is an optional parameter. If the *macaddr* is not coded, then the OSA-Express feature generates a virtual MAC address. If the *macaddr* value is coded, it must be defined as a locally administered individual MAC address. This means that the MAC address must have bit 6 (the universal or local flag U bit) of the first byte set to 1 and bit 7 (the group or individual flag G bit) of the first byte set to 0. The second hexadecimal character must be 2, 6, A, or E. The bit positions within the 12 hexadecimal characters are indicated in the following figure:

```
|              1|1              3|3              4|
|0             5|6              1|2              7|
+----------------+----------------+----------------+
|xxxxxxUGxxxxxxxx|xxxxxxxxxxxxxxxx|xxxxxxxxxxxxxxxx|
+----------------+----------------+----------------+
```

Rules:

- The same virtual MAC address generated by the OSA-Express feature at interface activation remains in effect for this OSA-Express for this TCP/IP stack, even if the device is stopped or becomes inoperative (INOPs). A new

virtual MAC address is generated only if the LINK statement is deleted and redefined, or if the TCP/IP stack is recycled.
- The NONROUTER, PRIROUTER, and SECROUTER parameters are ignored for an OSA-Express feature if the VMAC parameter is configured on the LINK statement.

Guideline: Unless the virtual MAC address representing this OSA-Express device must remain the same even after TCP/IP termination and restart, configure VMAC without a *macaddr* value and allow the OSA-Express device to generate it. This guarantees that the VMAC address is unique from all other physical burned-in MAC addresses and from all other VMAC addresses generated by any OSA-Express feature.

ROUTEALL
All IP traffic destined to the virtual MAC is forwarded by the OSA-Express device to the TCP/IP stack. This is the default value. See the router information in z/OS Communications Server: IP Configuration Guide for more details.

ROUTELCL
This specifies that only traffic destined to the virtual MAC and whose destination IP address is registered with the OSA-Express device by this TCP/IP stack is forwarded by the OSA-Express device. See the router information in z/OS Communications Server: IP Configuration Guide for more details.

Syntax

Rule: The optional parameters on the LINK statement following *device_name* can be specified in any order.

```
                                            ┌─NONCANONical─┐  ┌─ALLRINGsbcast─┐
►►──LINK──link_name──IPAQTR──device_name──┼──────────────┼──┼───────────────┼──────►
                                            └─CANONical────┘  └─LOCALBcast────┘

            ┌─READSTORAGE GLOBAL──────┐ ┌─INBPERF BALANCED──────────┐
►──┬─────────┬──┼─────────────────────────┼─┼───────────────────────────┼──────────►
   └─IPBCAST─┘  └─READSTORAGE──┬─MAX─┬─┘ └─INBPERF──┬─MINCPU─────┬─┘
                               ├─AVG─┤              └─MINLATENCY─┘
                               └─MIN─┘

   ┌─IFSPEED 100000000───────┐ ┌─SECCLASS 255────────────┐ ┌─NOMONSYSPLEX─┐
►──┼─────────────────────────┼─┼─────────────────────────┼─┼──────────────┼──────►◄
   ├─IFSPEED── ifspeed───────┤ └─SECCLASS security_class─┘ └─MONSYSPLEX───┘
   └─IFHSPEED── ifhspeed─────┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters.

IPAQTR
Indicates that the link uses the interface based on IP assist, belongs to the QDIO family of interfaces, and uses the Token Ring protocol.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

CANONICAL | NONCANONICAL
Specifies the canonical form of MAC addresses in Address Resolution Protocol (ARP) packets on this token-ring network.

NONCANONICAL
MAC addresses in ARP packets on this token-ring network are in the more common non-canonical format. This is the default value.

CANONICAL
MAC addresses in Address Resolution Protocol (ARP) packets on this token-ring network are in the canonical IEEE 802.5 form.

ALLRINGSBCAST
All IP and ARP broadcasts are sent as all-rings broadcasts, which are propagated through token-ring bridges (Source Route Bridging). This is the default value.

LOCALBCAST
All IP and ARP broadcasts are sent only on the local ring and are not propagated through token-ring bridges (Transparent Bridging).

IPBCAST
Specifies that the link both sends and receives IP broadcast packets. If this parameter is not specified, no IP broadcast packets are sent or received on this link.

READSTORAGE
An optional parameter indicating the amount of fixed storage that z/OS Communications Server should keep available for read processing for this adapter. The QDIOSTG VTAM start option allows you to specify a value which applies to all OSA-Express adapters in QDIO mode. You can use the READSTORAGE keyword to override the global QDIOSTG value for this adapter based on the inbound workload you expect over this adapter on this stack. The valid values are:

GLOBAL
The amount of storage is determined by the QDIOSTG VTAM start option. This is the default value.

MAX
Use this value if you expect a heavy inbound workload over this adapter

AVG
Use this value if you expect a medium inbound workload over this adapter.

MIN
Use this value if you expect a light inbound workload over this adapter.

Tip: See the description of the QDIOSTG VTAM start option in the z/OS Communications Server: SNA Resource Definition Reference for details about exactly how much storage is allocated by z/OS Communications Server for each of these values.

INBPERF
An optional parameter indicating how frequently the adapter should interrupt the host for inbound traffic. The valid values are:

MINCPU
This setting instructs the adapter to minimize host interrupts, thereby minimizing host CPU consumption. This mode of operation might result in minor queueing delays for packets into the host, and is not preferred for workloads with demanding latency requirements.

MINLATENCY
This setting instructs the adapter to minimize latency, by immediately presenting received packets to the host. This mode of operation generally results in higher CPU consumption than the other two settings, and is preferred only for workloads with demanding latency requirements. This setting should be used only if host CPU consumption is not an issue.

BALANCED
This setting instructs the adapter to strike a balance between MINCPU and MINLATENCY.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. Until the interface is successfully started, this value is used by SNMP as the value of the ifSpeed MIB object. After the interface is successfully started, SNMP uses the actual speed reported by the interface as the value of the ifSpeed MIB object. The value of this parameter has no effect on the operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. Until the interface is successfully started, this value is used by SNMP as the value of the ifHighSpeed MIB object. After the interface is successfully started, SNMP uses the actual speed reported by the interface as the value of the ifHighSpeed MIB object. The value of this parameter has no effect on the operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

Usage notes

- If using a dynamic VIPA when connecting an OSA-Express QDIO Ethernet device to an intelligent bridge or switch, ensure that the Spanning Tree Protocol (STP) on the intelligent bridge or switch is configured properly for dynamic VIPA giveback and takeover operations. For more information, see z/OS Communications Server: IP Configuration Guide.
- Across one central processor complex (CPC), PRIROUTER can be specified only in the profile of one TCP/IP instance for the same MPCIPA device. If PRIROUTER is specified for an MPCIPA device but was already specified for the same device in the profile of another active TCP/IP instance, a warning message is issued during START DEVICE processing for the device. Depending on the level of OSA-Express being started, either only one or multiple TCP/IP instances can be allowed to have SECROUTER specified. If OSA-Express allows only one secondary router, any TCP/IP instance subsequently starting that device with SECROUTER receives a warning message during START processing for the device. If OSA-Express allows multiple secondary routers, then OSA-Express can select any TCP/IP instance which specifies SECROUTER as the secondary router. There is no requirement that the same TCP/IP instance be specified PRIROUTER or SECROUTER for all OSA-Express adapters attached to the CPC.
- MPCIPA devices have an ARP off-load function that off-loads all ARP processing to the OSA-Express adapter. For some MPCIPA devices, TCP/IP cannot display any ARP cache information or ARP counter statistics because OSA-Express does not provide this data to TCP/IP.

  For more information about devices that support ARP Offload, see z/OS Communications Server: IP Configuration Guide.

Related topics

- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- “GATEWAY statement” on page 109
- “HOME statement” on page 136
- “INTERFACE - IPAQENET OSA-Express QDIO interfaces statement” on page 145
- “INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement” on page 161
- “SACONFIG statement” on page 271
- “START statement” on page 292
- “STOP statement” on page 293

DEVICE and LINK — MPCIPA HiperSockets devices statement

When defining an MPCIPA HiperSockets device, also known as an iQDIO device, use the DEVICE statement to specify the IQD CHPID hexadecimal value. The reserved device name using prefix IUTIQDxx must be specified. The suffix xx indicates the hexadecimal value of the corresponding IQD CHPID that was configured within HCD.

The hexadecimal value specified here cannot be the same value that is used for the dynamic XCF HiperSockets interface. See the IQDCHPID start option in z/OS Communications Server: SNA Resource Definition Reference.

MPCIPA HiperSockets devices do not require a corresponding TRLE. Instead, the TRLE is dynamically built when the device is started. There is no PORT name used for HiperSockets MPCIPA devices. The NONROUTER, PRIROUTER, and SECROUTER parameters do not apply to a HiperSockets device and are ignored if specified on the MPCIPA statement.

Use the LINK statement to define a network interface link associated with the HiperSockets interface.

Restriction: Only one LINK statement can be specified for each MPCIPA HiperSockets device.

Tip: You can also use the INTERFACE statement to define an IPv4 interface for HiperSockets, which combines the definitions of the DEVICE, LINK, and HOME statements into a single statement.

To determine the HiperSockets microcode level, use the DISPLAY TRL command. If a specific HiperSockets function is documented with a minimum microcode level, you can use this command to determine whether that function is supported. IBM service might request the microcode level for problem diagnosis. For more information, see DISPLAY TRL command in z/OS Communications Server: SNA Operation.

Syntax

Rule: Specify the parameters in the order shown here.

```
                                   ┌─NOAUTORestart─┐
►►──DEVice──device_name──MPCIPA────┼───────────────┼──────────────────────────────►◄
                                   └─AUTORestart───┘
```

Parameters

device_name
: The name of the device must use the following convention:
  - Prefix is IUTIQD.
  - Suffix xx [hexadecimal value (00x - FFx) of the corresponding IQD CHPID]. This value cannot conflict with the IQD CHPID used for dynamic XCF.

MPCIPA
: Specifies the device belongs to the MPC family of interfaces and uses the interface based on IP assist.

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

  NOAUTORESTART
  : For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
: In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see "Recovering from device failures" on page 48.

Syntax

Rule: The optional parameters on the LINK statement following *device_name* can be specified in any order.

```
                                                        ┌─READSTORAGE GLOBAL──┐
►►──LINK──link_name──IPAQIDIO──device_name──┬─────────┬─┼─────────────────────┼─┬───────────┬──────────►
                                            └─IPBCAST─┘ └─READSTORAGE──┬─MAX─┬┘ └─VLANIDid──┘
                                                                       ├─AVG─┤
                                                                       └─MIN─┘

  ┌─SECCLASS 255──────────────┐  ┌─NOMONSYSPLEX─┐
►─┼───────────────────────────┼──┼──────────────┼─────────────────────────────────────────────────────►◄
  └─SECCLASS security_class───┘  └─MONSYSPLEX───┘
```

Parameters

link_name
: The name of the link. The maximum length is 16 characters.

IPAQIDIO
: Indicates that the link uses the interface based on IP assist, belongs to the QDIO family of interfaces, and uses the HiperSockets protocol.

device_name
: The *device_name* must be the same as specified in the DEVICE statement.

IPBCAST
: Specifies that the link both sends and receives IP broadcast packets. If this parameter is not specified, no IP broadcast packets are sent or received on this link.

READSTORAGE
: An optional parameter indicating the amount of fixed storage that z/OS Communications Server should keep available for read processing for this device. The IQDIOSTG VTAM start option allows you to specify a value which applies to all HiperSockets devices. You can use the READSTORAGE keyword to override the global IQDIOSTG value for this adapter based on the inbound workload you expect over this device on this stack. The valid values are:

  GLOBAL
  : The amount of storage is determined by the IQDIOSTG VTAM start option. This is the default value.

  MAX Use this value if you expect a heavy inbound workload over this device.

  AVG Use this value if you expect a medium inbound workload over this device.

  MIN Use this value if you expect a light inbound workload over this device.

  Tip: See the description of the IQDIOSTG VTAM start option in the z/OS Communications Server: SNA Resource Definition Reference for details about exactly how much storage is allocated by z/OS Communications Server for each of these values.

Rule: If you define both a LINK and INTERFACE statement for the same device, then the READSTORAGE value on the LINK statement must match the READSTORAGE value on the corresponding INTERFACE statement. If you define a LINK statement that contains a value for READSTORAGE that conflicts with the READSTORAGE value for a previous INTERFACE statement for the same device, then TCP/IP rejects the LINK statement.

VLANID *id*

An optional parameter followed by a decimal number indicating the virtual LAN identifier to be assigned to this HiperSockets link. The valid range is 1 - 4094.

Restriction: With HiperSockets, a stack can specify only one VLAN ID when the interface is used for both IPv4 and IPv6. If you specify a different VLAN ID value on a LINK and INTERFACE definition for the same CHPID, the second statement is rejected.

SECCLASS *security_class*

Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX

Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX

Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX

Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

System environment

In order to configure a single HiperSockets device for both IPv4 and IPv6 traffic, you must use DEVICE/LINK/HOME for the IPv4 definition and INTERFACE for the IPv6 definition, such that the CHPID value on the INTERFACE statement matches the xx portion of the device_name (IUTIQDxx) on the DEVICE statement.

Related topics

- "BEGINROUTES statement" on page 28
- "BSDROUTINGPARMS statement" on page 36
- "GATEWAY statement" on page 109
- "HOME statement" on page 136
- "INTERFACE — IPAQIDIO HiperSockets interfaces statement" on page 157
- "INTERFACE — IPAQIDIO6 HiperSockets interfaces statement" on page 177
- "SACONFIG statement" on page 271
- "START statement" on page 292
- "STOP statement" on page 293

DEVICE and LINK — MPCOSA devices statement

When defining a multipath channel MPCOSA connection, use the DEVICE statement to specify the TRLE name of an HPDT connection. Use the LINK statement to specify Fast Ethernet OSA-2 or FDDI OSA-2.

Restrictions:

- MPCOSA devices cannot be configured to accept IP packets destined to an IP address other than the IP address of the OSA-2 adapter. For example, IP packets destined to a Virtual IP Address (VIPA) owned by this TCP/IP is not delivered by the OSA-2 adapter.
- MPCOSA devices do not support multicast or broadcast.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see "Missing interrupt handler factors" on page 49.

Syntax

Rule:

- Specify the parameters in the order shown here.

```
                                 ┌─NOAUTORestart─┐
►►──DEVice──device_name──MPCOSA──┴─AUTORestart───┴─────────────────────────►◄
```

Parameters

device_name
: For MPCOSA connections, the *device_name* must be the name of the TRLE definition that corresponds to the OSA-2 configuration. You need to use OSA/SF to configure the OSA-2 to run in HPDT MPC mode. The TRLE is defined in a VTAM TRL major node and must be active to start the device. For details about defining a TRLE, see z/OS Communications Server: SNA Resource Definition Reference.

  The maximum length is eight characters.

MPCOSA
: Specifies that the device is a multipath channel MCPOSA device.

AUTORESTART | NOAUTORESTART
Controls device failure reactivation behavior.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see "Recovering from device failures" on page 48.

Syntax

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

```
►►──LINK──link_name──┬─OSAFDDI─┬──link_number──device_name──────────────────►
                     └─OSAENET─┘


  ┌─IFSPEED 100000000──┐  ┌─SECCLASS 255────────────┐  ┌─NOMONSYSPLEX─┐
►─┼────────────────────┼──┼─────────────────────────┼──┼──────────────┼───────►◄
  ├─IFSPEED── ifspeed──┤  └─SECCLASS security_class─┘  └─MONSYSPLEX───┘
  └─IFHSPEED── ifhspeed┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters. The link name is associated with a home address on the HOME statement.

OSAFDDI
Specifies that the link is for MPCOSA FDDI OSA-2.

OSAENET
Specifies that the link is for MPCOSA Fast Ethernet OSA-2.

link_number
Specifies the OSA *link_number* of this interface and identifies the external attachment to a LAN supported by the OSA using the IP protocol.

Restriction: The only link number supported is 0.

device_name
The *device_name* must be the same as specified in the DEVICE statement. The maximum length is eight characters.

IFSPEED *ifspeed*
An optional estimate of the interfaces current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* is 0; the maximum value is 2 147 483 647. The default is 100 000 000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interfaces current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* is 0; the maximum value is 2147. The default is 100. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*

Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX

Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX

Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX

Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

Usage notes

- MPCOSA devices have an ARP offload function that offloads all ARP processing to the OSA-2 adapter. TCP/IP cannot display any ARP cache information or ARP counter statistics for these devices because OSA-2 does not provide this data to TCP/IP.
- To use dynamic routing with this device, see the NBMA subnetworks information in z/OS Communications Server: IP Configuration Guide.

Related topics

- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- “GATEWAY statement” on page 109
- “HOME statement” on page 136
- “START statement” on page 292
- “STOP statement” on page 293

DEVICE and LINK — MPCPTP devices statement

When defining a High Performance Data Transfer (HPDT) connection, use the DEVICE statement to specify the name of the TRLE definition for the multipath channel (MPC) group. Also, the TRLE must be defined as MPCLEVEL=HPDT.

When defining an Enterprise Extender connection to the VTAM instance running on this host, use the DEVICE statement to define an IUTSAMEH interface. IUTSAMEH can also be used to define a connection between two TCP/IP stacks on the same system, and the MPCPTP device and link statements can be used to define XCF connections between two TCP/IP stacks in the same sysplex. For more information about configuring Enterprise Extender, see z/OS Communications Server: SNA Network Implementation Guide.

Use the LINK statement to define a network interface link associated with an MPC group when defining an HPDT connection, or a network interface link associated with the IUTSAMEH interface when defining an Enterprise Extender connection.

The preferred way to define XCF and IUTSAMEH connections is to use the IPCONFIG DYNAMICXCF statement.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see “Missing interrupt handler factors” on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
                                  ┌─NOAUTORestart─┐
►►──DEVice──device_name──MPCPTP───┼───────────────┼─────────────────────────────►◄
                                  └─AUTORestart───┘
```

Parameters

device_name

For HPDT MPC connections to an IBM 2216 Multiaccess Connector Model 400, an IBM pSeries, or another z/OS host, the *device_name* must be the TRLE name of an HPDT connection. The TRLE is defined in a VTAM TRL major node and must be active to start the device. For details about defining a TRLE, see z/OS Communications Server: SNA Resource Definition Reference.

The maximum length is eight characters.

The reserved TRLE name IUTSAMEH can be used to bring up an MPCPTP connection between two TCP/IP stacks on the same system without the need for a physical device connection between the two stacks. The reserved TRLE name IUTSAMEH can also be used to define an Enterprise Extender connection to the VTAM instance running on this host. If you are defining an Enterprise Extender connection, the device name must be IUTSAMEH. VTAM automatically activates the IUTSAMEH TRLE.

For XCF connections, the *device_name* must be the CPname or SSCPname of the target VTAM on the other side of the XCF connection, and the VTAM ISTLSXCF major node must be active in both nodes to start the device. The ISTLSXCF major node is created by VTAM dynamically.

Tip: This value is also specified for *device_name* in the MPCPTP LINK statement.

MPCPTP
Specifies the device is a multipath channel point-to-point device.

AUTORESTART | NOAUTORESTART
Controls device failure reactivation behavior.

NOAUTORESTART
For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see "Recovering from device failures" on page 48.

Syntax

Rule: The optional parameters on the LINK statement following *device_name* can be specified in any order.

```
                                          ┌─CHECKSUM───┐
►►──LINK──link_name──MPCPTP──device_name──┼────────────┼──────────────────────────────►
                                          └─NOCHECKSUM─┘

   ┌─IFSPEED 4500000──────┐  ┌─SECCLASS 255────────────┐  ┌─NOMONSYSPLEX─┐
►──┼──────────────────────┼──┼─────────────────────────┼──┼──────────────┼──────────►◄
   ├─IFSPEED── ifspeed────┤  └─SECCLASS security_class─┘  └─MONSYSPLEX───┘
   └─IFHSPEED── ifhspeed──┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters. The link name is associated with a home address on the HOME statement.

MPCPTP
Specifies that the link is for MPCPTP.

device_name
The *device_name* must be the same as specified in the DEVICE statement. The maximum length is eight characters.

CHECKSUM
Inbound checksum calculation is performed for all packets received on this interface. This is the default value.

NOCHECKSUM
Inbound checksum calculation is not performed for any packets received on this interface.

The CHECKSUM or NOCHECKSUM setting affects only the inbound TCP/IP data path. This setting has no effect upon the outbound path (checksum calculation is always performed outbound).

While a performance gain can be achieved by specifying NOCHECKSUM, only specify NOCHECKSUM for single-hop MPCPTP links (that is, where application traffic terminates in the adjacent node), such as z/OS to pSeries point-to-point connections. In such a configuration, the z/OS channel provides a reliable data path, thereby minimizing the need for TCP/IP checksum in detecting transmission errors.

Guideline: The TCP/IP checksum is useful in detecting software errors at the sending side, so it is further suggested that NOCHECKSUM be specified only when the sending-side software is considered reliable.

Restriction: Do not specify NOCHECKSUM when the sending side is forwarding packets received over other devices. Systems forwarding packets do not check the transport layer (TCP or UDP) checksums; this is the responsibility of the final destination stack. In this case, disabling checksum processing can result in corrupted data being provided to the application.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. The minimum value for *ifspeed* is 0; the maximum value is 2 147 483 647. The default is 4 500 000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* is 0; the maximum value is 2147. The default is 4. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

Usage notes

Requirements:

- In order to configure a single physical device for both IPv4 and IPv6 traffic, you must use DEVICE/LINK/HOME for the IPv4 definition and INTERFACE for the IPv6 definition, such that the TRLENAME value on the INTERFACE statement matches the device_name on the DEVICE statement.
- IUTSAMEH definition is required if you plan to use the Enterprise Extender function and the TCP/IP stack you are configuring is used for access to the IP network by VTAM on this host.

Restriction: A mix of static and dynamic IPv4 and IPv6 definitions for a device is not allowed. For example, if a static IUTSAMEH IPv4 device and link is defined, an IPv6 dynamic definition for IUTSAMEH is created. If a static IUTSAMEH IPv6 interface is defined, an IPv4 dynamic definition for IUTSAMEH is not created. The same logic also applies for XCF links; a mix of static and dynamic IPv4 and IPv6 definitions is not allowed for an XCF link.

- If you start an MPCPTP device and the device does not become active and TCP/IP issues no messages in response to the start request, ensure that the remote end of this HPDT MPC connection is active. Even though the TRLE is active and a start device request was initiated, VTAM holds the TCP/IP start request waiting for the remote side of the HPDT MPC connection to become active.
- For installations that plan on dedicating the MPC group for exclusive use by a single TCP/IP stack, improved performance can be achieved by explicitly defining the MPC group as MPCUSAGE=EXC. For additional information about the MPCUSAGE keyword, see the z/OS Communications Server: SNA Resource Definition Reference.

Related topics

- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- For information about direct route restrictions, see “GATEWAY statement” on page 109.
- DYNAMICXCF in “IPCONFIG statement” on page 190
- “GATEWAY statement” on page 109
- “HOME statement” on page 136
- “INTERFACE — MPCPTP6 interfaces statement” on page 183
- “START statement” on page 292
- “STOP statement” on page 293

DEVICE and LINK — SNA LU0 links statement

| This device type will not be supported in a future release. It is recommended that
| you migrate to a later interface type, such as OSA-Express QDIO or HiperSockets.

Use the DEVICE statement to specify the name of the address space running the SNALINK program and the remote SNA LU name of the 3745 Communications Controller to which an Ethernet or token-ring is attached. These statements are required for NCPROUTE when using NCP V7R3 or earlier.

Use the LINK statement to define the link on the SNA LU type 0 DEVICE statement. Use this method to configure TCP/IP to access the 3745 adapter through SNALINK.

Syntax

Rule: Specify the parameters in the order shown here.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see "Missing interrupt handler factors" on page 49.

```
►►──DEVICE──device_name──SNAIUCV SNALINK──lu_name──proc_name──────────────────►

   ┌─NOAUTORestart─┐
►──┼───────────────┼──────────────────────────────────────────────────────────►◄
   └─AUTORestart───┘
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

SNAIUCV SNALINK
: Specifies that the connection operates as an SNA LU type 0.

lu_name
: The logical unit (LU) name of the remote end. The maximum length is eight characters.

proc_name
: The name of the SNALINK started procedure that runs on the host end. The maximum length is eight characters.

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

NOAUTORESTART
: For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
: In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see "Recovering from device failures" on page 48.

Syntax

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

```
►►──LINK──link_name──┬─SAMEHOST─┬──link_number──device_name──────────────────►
                     └─IUCV─────┘

   ┌─IFSPEED 56000───────┐  ┌─SECCLASS 255──────────────┐  ┌─NOMONSYSPLEX─┐
►──┼─────────────────────┼──┼───────────────────────────┼──┼──────────────┼────►◄
   ├─IFSPEED── ifspeed───┤  └─SECCLASS security_class──┘  └─MONSYSPLEX───┘
   └─IFHSPEED── ifhspeed─┘
```

Restriction: There must be only one LINK statement for each SNA LU type 0 device statement.

Parameters

link_name
The name of the link. The maximum length is 16 characters.

SAMEHOST
Specifies that the DEVICE for SNA LU type 0 support uses a SAMEHOST connection.

Note on IUCV: The IUCV keyword remains for migration purposes and is identical to SAMEHOST.

link_number
The *link_number* must be an integer, but its value is ignored. This parameter is included for consistency with LINK statement formats for other device types.

device_name
The *device_name* must be the same as specified in the DEVICE statement. The maximum length is 16 characters.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* is 0; the maximum value is 2147483647. The default is 56000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* is 0; the maximum value is 2147. The default is 0. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See "Modifying DEVICE and LINK statements" on page 50 for modifying information.

Examples

In this example, SNALU0 is an SNA Link.

```
DEVICE SNALU0 SNAIUCV SNALINK LU000000 SNALINK
LINK SNA1 SAMEHOST     1 SNALU0
```

Usage notes

You can specify multiple LU0 DEVICE statements for the same SNALINK started procedure. A single LU0 address space can support multiple SAMEHOST links. A SAMEHOST link is created for each pair of LU0 DEVICE and LINK statements.

However, you must specify a different *lu_name* for each DEVICE statement. This value is passed to the LU0 application to establish a session with a remote LU of that name.

Related topics

- "BEGINROUTES statement" on page 28
- "BSDROUTINGPARMS statement" on page 36
- "GATEWAY statement" on page 109
- "HOME statement" on page 136
- "START statement" on page 292
- "STOP statement" on page 293

DEVICE and LINK — SNA LU 6.2 links statement

| This device type will not be supported in a future release. It is recommended that
| you migrate to a later interface type, such as OSA-Express QDIO or HiperSockets.

Use the DEVICE statement to specify the name of the started procedure running the SNALINK LU 6.2 interface program.

Use the LINK statement to define the link to the SNALINK LU 6.2 Interface program.

Restriction: There must be only one LINK statement for each SNA LU type 6.2 DEVICE statement.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see "Missing interrupt handler factors" on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
                                              ┌─NOAUTORestart─┐
►►──DEVICE──device_name──SNALU62──proc_name───┼───────────────┼──────────────────►◄
                                              └─AUTORestart───┘
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

SNALU62
: Specifies that the connection operates by an SNA LU type 6.2 session.

proc_name
: The name of the SNALINK started procedure (on this node) that controls the device. The maximum length is eight characters.

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

  NOAUTORESTART
  : For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

  AUTORESTART
  : In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see "Recovering from device failures" on page 48.

Syntax

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

```
►►──LINK──link_name──┬─SAMEHOST─┬──link_number──device_name──────────────────────►
                     └─IUCV─────┘

   ┌─IFSPEED 56000───────┐  ┌─SECCLASS 255────────────┐  ┌─NOMONSYSPLEX─┐
►──┼─────────────────────┼──┼─────────────────────────┼──┼──────────────┼───────►◄
   ├─IFSPEED── ifspeed───┤  └─SECCLASS security_class─┘  └─MONSYSPLEX───┘
   └─IFHSPEED── ifhspeed─┘
```

Parameters

link_name
: The name of the link. The maximum length is eight characters. The same name is specified in the SNALINK LU 6.2 configuration data set (*hlq*.PROFILE.TCPIP) to identify this link.

SAMEHOST
: A constant that specifies that the device for SNA LU type 6.2 support uses a SAMEHOST connection.

  Note on IUCV: The IUCV keyword remains for migration purposes and is identical to SAMEHOST.

link_number
: The *link_number* must be an integer, but the value is ignored. This parameter is included for consistency with LINK statement formats for other device types.

device_name
: The *device_name* must be the same as specified in the DEVICE statement. The maximum length is 16 characters.

IFSPEED *ifspeed*
: An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* is 0; the maximum value is 2 147 483 647. The default is 56 000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
: An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* is 0; the maximum value is 2 147. The default is 0. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
: Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

  Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

  Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
: Specifies whether or not sysplex autonomics should monitor the link's status.

  NOMONSYSPLEX
  : Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

  MONSYSPLEX
  : Specifies that sysplex autonomics should monitor the link's status.

    Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

Related topics

- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- “GATEWAY statement” on page 109
- “HOME statement” on page 136
- “START statement” on page 292
- “STOP statement” on page 293

DEVICE and LINK — VIRTUAL devices statement

Use the DEVICE statement to specify the device name of a static virtual device, and use the LINK statement to define the link on the DEVICE statement.

More than one virtual DEVICE/LINK statement can be defined to allow for multiple virtual IP addresses on one TCP/IP image in one MVS system.

Tip: You can also use the INTERFACE statement to define an IPv4 interface for a static VIPA, which combines the definitions of the DEVICE, LINK, and HOME statements into a single statement.

This statement applies to IPv4. See “INTERFACE — VIRTUAL6 interfaces statement” on page 188 for this function in IPv6.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──DEVice──device_name──VIRTual──device_number──────────────────────────►◄
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

VIRTUAL
: Specifies that the device is not associated with real hardware and is used for fault tolerance support. The static virtual devices always stay active and are never subject to physical failure.

device_number
: The *device_number* must be a hexadecimal number, but the value is ignored. This parameter is included for consistency with the DEVICE statements for other device types.

Syntax

```
►►──LINK──link_name──VIRTual──adapter_address──device_name──────────────────────►◄
```

Restriction: Only one LINK statement can be defined for each virtual device.

Parameters

link_name
The name of the link. The maximum length is 16 characters. The same name is specified in the HOME statement.

VIRTUAL
Specifies that the link is a virtual link that is not associated with real hardware and is used for fault tolerance support.

adapter_address
The *adapter_address* must be an integer, but the value is ignored. This parameter is included for consistency with the LINK statements for other device types.

device_name
The *device_name* must be the same as specified in the DEVICE statement.

Steps for modifying

See "Modifying DEVICE and LINK statements" on page 50 for modifying information.

Guideline: The steps in the Modifying DEVICE and LINK statements topic that refer to stopping and starting the device do not apply to virtual devices.

Examples

```
DEVICE VDEV1  VIRTUAL 0
LINK   VLINK1 VIRTUAL 0 VDEV1
DEVICE VDEV2  VIRTUAL 1
LINK   VLINK2 VIRTUAL 0 VDEV2
```

Usage notes

- The *device_name* or *link_name* values should not start with VIP because VIP is a restricted keyword.
- A virtual LINK cannot be coded on the START, BEGINROUTES, GATEWAY or TRANSLATE statements, but can be coded on a BSDROUTINGPARMS statement for interface characteristics such as subnet mask.

Requirement: If you are running with 3172s configured for multihost connectivity (release 3.5 and later) and want to use VIPA addresses on the host, you must configure the 3172 in one of the following ways:
- As a default router (routes all IP addresses)
- Configure all VIPA addresses in the 3172
- For rules on defining virtual IP addresses for virtual links, see "HOME statement" on page 136.

Related topics

- "BSDROUTINGPARMS statement" on page 36
- "HOME statement" on page 136

- “INTERFACE — VIRTUAL interfaces statement” on page 160
- “IPCONFIG statement” on page 190
- “VIPADYNAMIC statement summary” on page 306

DEVICE and LINK - X.25 NPSI connections statement

This device type will not be supported in a future release. It is recommended that you migrate to a later interface type, such as OSA-Express QDIO or HiperSockets.

Use the DEVICE statement to specify the name and address of the X.25 NPSI interface program devices that you use. Use the LINK statement to define a network interface link associated with the X.25 NPSI interface program devices.

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see “Missing interrupt handler factors” on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
                                                ┌─NOAUTORestart─┐
►►──DEVICE──device_name──X25NPSI──proc_name──┼───────────────┼────────────────►◄
                                                └─AUTORestart───┘
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

X25NPSI
: Specifies that the device is an X.25 NPSI.

proc_name
: The name of the X.25 NPSI server started procedure. The maximum length is eight characters.

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

NOAUTORESTART
: For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
: In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see “Recovering from device failures” on page 48.

Restriction: Only one DEVICE and LINK statement per TCPIPX25 address space is allowed.

Syntax

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

```
►►──LINK──link_name──┬─SAMEHOST─┬──link_number──device_name──────────────────►
                     └─IUCV─────┘

   ┌─IFSPEED 56000──────┐   ┌─SECCLASS 255────────────┐   ┌─NOMONSYSPLEX─┐
►──┼────────────────────┼───┼─────────────────────────┼───┼──────────────┼──────►◄
   ├─IFSPEED── ifspeed──┤   └─SECCLASS security_class─┘   └─MONSYSPLEX───┘
   └─IFHSPEED── ifhspeed┘
```

Parameters

link_name
The name of the link. The maximum length is 16 characters.

SAMEHOST
Specifies that the connection to X.25 NPSI is established using a SAMEHOST connection.

Note on IUCV: The IUCV keyword remains for migration purposes and is identical to SAMEHOST.

link_number
The *link_number* must be an integer, but its value is ignored. This parameter is included for consistency with LINK statement formats for other device types.

device_name
The *device_name* must be the same as specified in the DEVICE statement. The maximum length is 16 characters.

IFSPEED *ifspeed*
An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* is 0; the maximum value is 2 147 483 647. The default is 56 000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* is 0; the maximum value is 2 147. The default is 0. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

Examples

This example shows how you might code DEVICE, LINK, and related statements for an X.25 connection.

```
DEVICE X25DEV X25NPSI TCPIPX25
LINK X25LINK SAMEHOST 1 X25DEV
;
HOME
   199.005.058.23     X25LINK
;
GATEWAY
;
; Network   First Hop   Link name   Packet size   Subnet mask   Subnet Value
  192.005   =           X25LINK     2000          0.0.255.0     0.0.58.0
;
START X25DEV
;
```

Usage notes

To use dynamic routing with this device, see the NBMA subnetworks information in z/OS Communications Server: IP Configuration Guide.

Related topics

- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- “GATEWAY statement” on page 109
- “HOME statement” on page 136
- “START statement” on page 292
- “STOP statement” on page 293
- Chapter 16, “TN3270E Telnet server,” on page 587

DEVICE and LINK — 3745/46 channel DLC devices statement

| This device type will not be supported in a future release. It is recommended that
| you migrate to a later interface type, such as OSA-Express QDIO or HiperSockets.

Use the DEVICE statement to specify the name and hexadecimal device number of the channel data link control (CDLC) devices that you use. Use the LINK statement to define a network interface link associated with the CDLC devices.

If the device is running NCP V7R3 or earlier and dynamic routing is to be performed, SNALINK must be configured to carry RIP transport PDUs:

- NCP V7R3 or earlier does not support native IP transmission across the channel of the transport PDUs associated with RIP traffic (NCP V7R3 or earlier expects these PDUs to be carried in SNA Frames). SNALINK is still required in environments where dynamic routing is performed with the NCP V7R3 or earlier (using NCPROUTE).

  To minimize the amount of data sent across the SNALINK (LUO) connection (as SNALINK consumes more CPU than does IP over CDLC), use the RIP Filter to send RIP updates across the channel, while the associated transport PDUs (Route Table Management, for example, Handshaking, Add Route Request, Delete Route Request) are carried over the SNALINK connection.
- If the device is running a later version than NCP V7R3, or if the device is a 3746 model 950, SNALINK is not required (all IP and RIP traffic can be transported over direct CDLC link of TCP/IP).

For more information about missing interrupt handler (MIH) considerations with TCP/IP devices, see “Missing interrupt handler factors” on page 49.

Syntax

Rule: Specify the parameters in the order shown here.

```
                                                  ┌─15───────────┐
►►──DEVICE──device_name──CDLC──device_number──────┼──────────────┼────────────────────►
                                                  └─read_buffers─┘

   ┌─15────────────┐   ┌─4096──────┐   ┌─4096───────┐   ┌─NOAUTORestart─┐
►──┼───────────────┼───┼───────────┼───┼────────────┼───┼───────────────┼───────────►◄
   └─write_buffers─┘   └─read_size─┘   └─write_size─┘   └─AUTORestart───┘
```

Parameters

device_name
: The name of the device. The maximum length is 16 characters. The same name is specified in the LINK statement.

CDLC
: Specifies that this device is to run the CDLC protocol.

device_number
: The hexadecimal device number (in the range 0 - FFFF) of the CDLC device.

read_buffers
: The decimal number of buffers to allocate to the read channel program. The default is 15. The minimum is 1 and the maximum is 63. The product of *read_buffers* times *read_size* must be less than or equal to 65535. If the product of these configured variables exceeds 65535, TCP/IP reduces *read_buffers* to the integer 65535/read_size.

write_buffers
: The decimal number of buffers to allocate to the write channel program. The minimum is 1 and the maximum is 63. The product of *write_buffers* times

write_size must be less than or equal to 65535. The default is 15. If the product of these configured variables exceeds 65535, TCP/IP reduces *write_buffers* to the integer 65535/write_size.

read_size
: The size in bytes (decimal) of the read buffers. The default is 4096. Valid values are 1024, 2048, 4096, 6144, 8192.

write_size
: The size in bytes (decimal) of the write buffers. The default is 4096. Valid values are 1024, 2048, 4096, 6144, 8192.

AUTORESTART | NOAUTORESTART
: Controls device failure reactivation behavior.

NOAUTORESTART
: For most device failures, specifying NOAUTORESTART indicates that the TCP/IP address space does not attempt to reactivate this device.

AUTORESTART
: In the event of a device failure, the TCP/IP address space attempts to reactivate the device. For more information, see "Recovering from device failures" on page 48.

Syntax

Rule: The optional parameters on the LINK statement following the *device_name* parameter can be specified in any order.

```
                                                  ┌─IFSPEED 4500000──────┐
►►──LINK──link_name──CDLC──adapter_addr──device_name─┼──────────────────────┼──────────►
                                                  ├─IFSPEED── ifspeed────┤
                                                  └─IFHSPEED── ifhspeed──┘

  ┌─SECCLASS 255──────────────┐   ┌─NOMONSYSPLEX─┐
►─┼───────────────────────────┼───┼──────────────┼──────────────────────────────►◄
  └─SECCLASS security_class───┘   └─MONSYSPLEX───┘
```

Parameters

link_name
: The name of the link. The maximum length is 16 characters.

CDLC
: Specifies that the link is a channel DLC.

adapter_addr
: The *adapter_addr* value must be an integer, but the value is ignored. This parameter is included for consistency with the LINK statement formats for other device types.

device_name
: The *device_name* must be the same as specified in the DEVICE statement.

IFSPEED *ifspeed*
: An optional estimate of the interface's current bandwidth in bits per second. The minimum value that can be specified for *ifspeed* is 0; the maximum value is 2147483647. The default is 4500000. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

IFHSPEED *ifhspeed*
An optional estimate of the interface's current bandwidth in one million bits per second units. The minimum value that can be specified for *ifhspeed* is 0; the maximum value is 2147. The default is 4. This value is accessible to SNMP for management queries, but has no effect on operation of the device.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide..

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the link's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the link's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the link's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this link is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Modifying DEVICE and LINK statements” on page 50 for modifying information.

Usage notes

For a buffer size of 8192, the maximum number of buffers is 7. For a buffer size of 6144, the maximum number of buffers is 10. For a buffer size of 4096, the maximum number of buffers is 15. For a buffer size of 2048, the maximum number of buffers is 31. For a buffer size of 1024, the maximum number of buffers is 63.

Related topics

- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- “GATEWAY statement” on page 109
- “HOME statement” on page 136
- “STOP statement” on page 293
- “START statement” on page 292

GATEWAY statement

The GATEWAY statement will not be supported in a future release. It is recommended that you use the BEGINROUTES/ENDROUTES statement block instead of the GATEWAY statement.

Tip: To use IPCS to create BEGINROUTES statements from the existing specification, run the TCPIPCS PROFILE report on a dump of the TCP/IP address space. Static routes are presented as a BEGINROUTES/ENDROUTES statement block even if you code the static routes by using a GATEWAY statement.

Use the GATEWAY statement to add static routes to the main route table. If you want to specify static routes in a BSD style syntax or if you need to define routes for interfaces defined with the INTERFACE statement, see “BEGINROUTES statement” on page 28. To configure policy-based route tables, use the RouteTable statement. For more information, see the policy-based routing information in z/OS Communications Server: IP Configuration Guide.

IBM Health Checker for z/OS can be used to check whether the total number of indirect routes in a TCP/IP stack routing table exceeds a maximum threshold. When this threshold is exceeded, OMPROUTE and the TCP/IP stack may potentially experience high CPU consumption from routing changes. A large routing table is considered to be inefficient in network design and operation. For more details about IBM Health Checker for z/OS, see z/OS Communications Server: IP Diagnosis Guide and IBM Health Checker for z/OS: User's Guide..

The IP static routes can be modified in the following ways:
- Replace the routing table using the VARY TCPIP,,OBEYFILE command.
- Use incoming ICMP redirect packets if redirects have not been disabled on the IPCONFIG statement.

The first GATEWAY statement of each configuration data set that is issued replaces all the static routes in the existing routing table with the new gateway information. All static routes and any dynamic routes added as a result of ICMP redirects are deleted. Routes created by OMPROUTE are not deleted. Subsequent GATEWAY statements in the same data set add entries to the routing table.

Restrictions:
- The GATEWAY statement applies only to IPv4 interfaces that are defined with the LINK statement.
- A GATEWAY statement and a BEGINROUTES-ENDROUTES block cannot be intermixed in the same configuration data set. If they are intermixed, the first type found is used and the other type is discarded with warning messages being issued to the console. You can use a GATEWAY statement in the initial profile and a BEGINROUTES-ENDROUTES block in the data set specified on a later VARY TCPIP,,OBEYFILE command, and vice versa.
- Route entries on the GATEWAY statement can be coded only for LINK names that exist when the entry is processed.
- Static routes that you define using the GATEWAY statement cannot be replaced by dynamic routes learned by OMPROUTE. If you want OMPROUTE to begin managing all routes, use an empty GATEWAY statement block in a VARY TCPIP,,OBEYFILE command data set to eliminate the existing static routes.

OMPROUTE discovers the changes dynamically. If you want to define static routes that can be replaced by dynamic routes learned by OMPROUTE, use the BEGINROUTES statement.
- VIPA links are not allowed on the GATEWAY statement.

Note:

1. When an incorrect entry in a GATEWAY statement is encountered, it is discarded along with the remaining entries in the GATEWAY statement. All routes defined before the incorrect entry are added to the main route table. Subsequent GATEWAY statements in the same profile data set or in the VARY TCPIP,,OBEYFILE command data set are processed.
2. A specific host route takes precedence over a subnetwork route, followed by a network route, followed by a supernetwork route, and finally, a default route.
3. To add a route over a MPCPTP link to another IP address (for example a VIPA) on the remote host, you need to add an indirect route. This indirect route should have a destination equal to the other IP address, a *first_hop* value equal to the remote IP address of the MPCPTP link, and a *link_name* value equal to the name of the MPCPTP link.

Rule: Specify the required parameters in the order shown here. The optional parameters can be specified in any order.

Syntax

```
                 ┌────────────────────────┐
                 ▼                        │
►►──GATEway──────┤ Gateway List Entry ├───┴────────────────────────────────►◄
```

Gateway List Entry:

```
├──┬─┤ Gateway Network Specification ├─────────┬──────────────────────────►
   ├─┤ Gateway Host Specification ├────────────┤
   └─┤ Gateway Default Network Specification ├─┘

►─┤ Gateway List Entry Options ├────────────────────────────────────────────┤
```

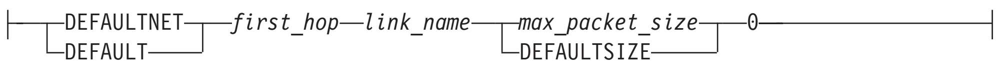
Gateway Network Specification:

```
├──network──┬─first_hop─┬──link_name──┬─max_packet_size─┬──────────────────►
            └─ = ───────┘             └─DEFAULTSIZE─────┘

►─┬─subnet_mask──subnet_value─┬─────────────────────────────────────────────┤
  └─0─────────────────────────┘
```

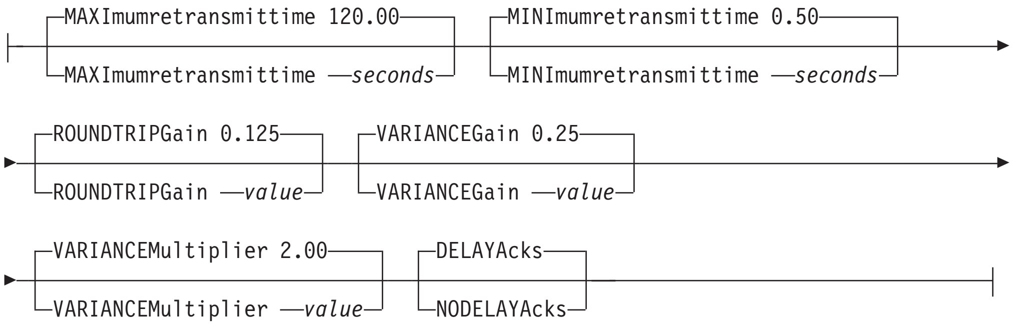
Gateway Host Specification:

```
├──host──┬─first_hop─┬──link_name──┬─max_packet_size─┬───HOST──────────────┤
         └─ = ───────┘             └─DEFAULTSIZE─────┘
```

Gateway Default Network Specification:



Gateway List Entry Options:


MAXImumretransmittime 120.00
MAXImumretransmittime —seconds
MINImumretransmittime 0.50
MINImumretransmittime —seconds
ROUNDTRIPGain 0.125
ROUNDTRIPGain —value
VARIANCEGain 0.25
VARIANCEGain —value
VARIANCEMultiplier 2.00
VARIANCEMultiplier —value
DELAYAcks
NODELAYAcks

Parameters

network
The IP address in dotted decimal form.

- An example of a class A network is 9.0.0.0.
- An example of a class B network is 129.34.0.0.
- An example of a class C network is 192.9.100.0.

Use the *subnet_mask* and *subnet_value* fields to completely define the route. Multiple network routes having an identical destination IP address and address mask can be specified. When multiple routes are specified, all of them are used when multipath is enabled on the IPCONFIG statement; otherwise, only the first active route specified is used.

first_hop
Specify one of the following values:

- An equal sign (=), meaning that datagrams are routed directly to destinations on that network or directly to that host over the interface that is identified by *link_name*. This is not supported for DEFAULT or DEFAULTNET.
- The IP address of a gateway or router that you can reach directly, and that forwards datagrams for the destination network or host over the interface that is identified by *link_name*. The address must be a host address that uniquely identifies the gateway or router. It cannot be a local IP address on this TCP/IP stack. A local IP address can be defined on the HOME, INTERFACE, VIPADEFINE, or IPCONFIG DYNAMICXCF statement. The IP address must be a fully qualified address in the form `a.b.c.d`.

link_name
The name of the link through which packets are sent to the specified network. The link name is defined in a LINK statement.

max_packet_size
The maximum transmission unit (MTU) in bytes for the network or host. This value can be up to 65535.

See Figure 1 on page 48 for more information about the largest MTU value supported by each link type.

The DEFAULTSIZE keyword sets the *max_packet_size* to 576. This value is the minimum MTU that an IPv4 network should use. See Usage notes for packet size considerations.

Tip: See z/OS Communications Server: IP Configuration Guide, under section, Maximum transmission unit considerations, for additional information about how TCP/IP uses the MTU to determine the largest size frame to send.

subnet_mask
A bit mask (expressed in dotted decimal form) having bits in the network portions or host portions, or both, that defines the subnet mask associated with the route. If a route to the network is not subnetted, specify a subnet mask of 0 and omit the subnet value. For a specific host route, specify a subnet mask of HOST and omit the subnet value.

Restriction: The mask bits of all ones in the host portion cannot be used for the subnet mask.

The valid values are:
- A dotted decimal bit mask
- 0
- Host

subnet_value
Value of the subnet. Each subnet should have a unique dotted decimal representation. Do not include the *subnet_value* field if the *subnet_mask* is 0, HOST, or contains a supernet mask.

If the network has one or more subnets, specify a separate entry in the GATEWAY statement for each subnet. The network part of each GATEWAY entry is identical (contains the IP network address as if the network has no subnets). The *subnet_mask* part of each GATEWAY entry might be identical, but the *subnet_value* varies.

host
The host address, specified as 4 octets (192.9.100.3, for example). If a host address is specified, the keyword HOST must be specified in place of the *subnet_mask* value, and the *subnet_value* value must be omitted. Multiple host routes having an identical destination IP addresses and address masks can be specified. When multiple routes are specified, all of them are used when multipath is enabled on the IPCONFIG statement; otherwise, only the first active route specified is used.

DEFAULTNET
Specifies a route to use for any destination that is not covered by an explicit route. You can specify multiple DEFAULTROUTE entries to provide multiple default routes. When you specify multiple routes, all of them are used when multipath is enabled on the IPCONFIG statement; otherwise, only the first active route that is specified is used. The DEFAULT and DEFAULTROUTE parameters are interchangeable.

DEFAULT
Specifies a route to use for any destination that is not covered by an explicit route. You can specify multiple DEFAULT entries to provide multiple default routes. When you specify multiple routes, all of them are used when multipath is enabled on the IPCONFIG statement; otherwise, only the first active route that is specified is used. The DEFAULT and DEFAULTROUTE parameters are interchangeable.

Retransmission parameter considerations

The parameters listed in this topic affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a specified number of retransmits, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times that packets are retransmitted before the connection times out differs for initial connection establishment and for data packets . For initial connection establishment, the initial time interval is set at approximately 3 seconds and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval, and data packets are retransmitted 15 times before the connection is timed out. All of the remaining parameters listed in this topic affect the data packet retransmission algorithm. Only the MINIMUMRETRANSMITTIME parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

The retransmission parameters enable system administrators who are familiar with TCP/IP transmission performance to alter the flow of TCP/IP data packets and acknowledgments. Under normal circumstances, the following occurs:
- TCP typically waits to receive two packets before sending one ACK to acknowledge the data within them.
- When TCP sends a packet, it waits for an acknowledgment. If it times out before getting an acknowledgment, it resends the packet.

Use the following parameters to adjust the retransmission time-out calculations; slower transmission times prevent packets from being resent as quickly:
- MAXIMUMRETRANSMITTIME
- MINIMUMRETRANSMITTIME
- ROUNDTRIPGAIN
- VARIANCEGAIN
- VARIANCEMULTIPLIER
- DELAYACKS
- NODELAYACKS

TCP uses these values in an algorithm called the TCP Retransmission Timeout Calculation, which is described in RFC 793. When you use this calculation, the following occurs:
- A smoothed round trip time (SRTT) and variance (VAR) is updated from the individual RTT derived from each packet acknowledgement.
- The retransmit time for a new packet is set to twice (approximately) the current SRTT value plus the VAR value.
- Each time a packet is retransmitted, the retransmit time value is doubled.

- The actual interval time used for the initial packet and each retransmission is the retransmit time calculated previously, but limited by the configured MINIMUMRETRANSMITTIME and MAXIMUMRETRANSMITTIME values.

DELAYACKS | NODELAYACKS
Controls transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header.

NODELAYACKS
Specifies that an acknowledgment is returned immediately when a packet is received with the PUSH bit on in the TCP header. The NODELAYACKS parameter on the BEGINROUTES, GATEWAY, and RouteTable statements affects only the connections that use this route. Specifying NODELAYACKS on the TCP/IP stack BEGINROUTES or GATEWAY profile statements, or on the Policy Agent RouteTable statement, overrides the specification of the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

DELAYACKS
Delays transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. The DELAYACKS parameter on the BEGINROUTES, GATEWAY, and RouteTable statements affects only the connections that use this route. This is the default, but you can override the default by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements.

MAXIMUMRETRANSMITTIME
Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to timeout. Specifying MAXIMUMRETRANSMITTIME assures that the interval time never exceeds the specified limit. The minimum value that can be specified for MAXIMUMRETRANSMITTIME is 0. The maximum is 999.990. The default is 120 seconds. This parameter does not affect initial connection retransmission.

MINIMUMRETRANSMITTIME
Sets a minimum retransmit interval. Increasing this value might increase the amount of time it takes for TCP to time out a connection. The minimum value that can be specified for MINIMUMRETRANSMITTIME is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

ROUNDTRIPGAIN
This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet RTT has on the average. The minimum value that can be specified for ROUNDTRIPGAIN is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

VARIANCEGAIN
This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum value that can be specified for VARIANCEGAIN is 0. The maximum value is 1.0. The default is 0.25. This parameter does not affect initial connection retransmission.

VARIANCEMULTIPLIER
This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT

has on calculating the retransmission interval. The minimum value that can be specified for VARIANCEMULTIPLIER is 0. The maximum value is 99.990. The default is 2. This parameter does not affect initial connection retransmission.

Retransmission parameters

Use the ROUNDTRIPGAIN, VARIANCEGAIN, and VARIANCEMULTIPLIER parameters to instruct TCP how heavily to weigh the most recent behavior of the network versus the long term behavior for updating the SRTT and VAR values. If you specify smaller values for these parameters, TCP attempts to correct for congestion only if the congestion is sustained. With larger values, TCP corrects for congestion more quickly, and the system is more sensitive to variations in network performance. Use the default values (unless your retransmission rate is too high).

Use DELAYACKS to delay the acknowledgments so that they can be combined with data sent to the foreign host.

Steps for modifying

To modify any values on the GATEWAY statement, use a VARY TCPIP,,OBEYFILE command with a data set that contains a new GATEWAY statement. All existing static routes are deleted along with all routes learned by way of ICMP redirects, but routes created by OMPROUTE are not deleted. To remove all static routes from the main routing table, specify an empty GATEWAY statement.

Note:
1. If any HOME list entries are changed or deleted, all static routes using the associated links are deleted.
2. If a LINK becomes inactive, then all routes that are associated with that link are marked inactive.
3. If a LINK becomes active, then all static routes that are associated with that link are marked active.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands.

Usage notes

- **Packet size considerations**

  The actual packet size is determined by the total network connection.
  - The *max_packet_size* varies for different networks. For example, the largest packet size for the Ethernet protocol network is 1500 bytes while the largest packet size for the IEEE 802.3 protocol network is 1492 bytes.
  -
    - If a locally attached host has a packet size smaller than yours, transfers to that host use the smaller size.
    - The TCP maximum segment size for the 3172 Interconnect Controller Program is 4096. Any packet specifications over 4096 are limited by this restriction. For example, if you specified a packet size of 4352, the resulting packet size would still only be 4096 + the header = 4132.
  - Large packets can be fragmented by intervening gateways. Fragmentation and reassembly of packets are expensive in their use of bandwidth and CPU time. Therefore, packets sent through gateways to other networks should use the default size, DEFAULTSIZE, unless one of the following situations is true:

- All intervening gateways and networks are known to accept larger packets
- Path MTU discovery is enabled on the IPCONFIG statement, which results in the TCP/IP stack dynamically learning the maximum MTU for the total network connection.

For example, the router to network 192.8.4 is reached through router 14.0.0.10, and somewhere along the path, packets larger than 576 bytes are fragmented. You can improve throughput by using the following GATEWAY statement:

```
GATEWAY
; Network   first-hop   link   packet-size    subnet-mask
  192.8.4   14.0.0.10   LINK1      576            0
```

– You cannot specify an MTU smaller than the default MTU size. For IPv4, the default MTU is 576.

- **Requirement:** The subnet_mask must follow the Classless Inter-Domain Routing (CIDR) convention that requires the actual mask to be one or more on-bits followed by zero or more off-bits. You cannot have on-bits followed by off-bits followed by on-bits. Therefore, a class A subnet mask of 0.255.254.0 is valid (an actual mask of 255.255.254.0 or FFFFE00), but a class A subnet mask of 0.255.253.0 is not valid (an actual mask of 255.255.253.0 or FFFFD00) because 253 is 11111101.
- The first GATEWAY statement of each profile data set or VARY TCPIP,,OBEYFILE command data set deletes only IPv4 routes. Any IPv6 static routes added previously are not deleted. The IPv6 routes must have been added using the BEGINROUTES statement.
- If the routing table is empty, all addresses in the HOME list are still route-capable. For information about testing commands with LOOPBACK, see the z/OS Communications Server: IP User's Guide and Commands.
- There is no limit on the number of equal-cost multipath routes to a destination.
- Multicast routes can be specified using host specification. A general multicast default route can be specified using the multicast group address of 224.0.0.0:

```
GATEWAY

;Host      First hop  Link   packet size

 224.0.0.0  =         LINK1  DEFAULTSIZE  HOST
```

Specific multicast group routes can also be specified:

```
GATEWAY

;Host      First hop  Link   packet size

 224.1.1.1  =         LINK2  DEFAULTSIZE  HOST
```

The order of precedence for determining the route of an outbound multicast datagram is as follows:

1. Application uses setsockopt() IP_MULTICAST_IF to specify the interface to use.
2. The specific multicast group route that is specified.
3. Multicast network or prefix route.
4. The general multicast default group address that is specified (224.0.0.0).
5. If a default route is specified and the route's link is multicast capable.

Given the preceding two sample GATEWAY statements and assuming the application does not code the setsockopt() IP_MULTICAST_IF, one of the following situations occurs:

– If an application sends a datagram to 224.2.2.2, LINK1 is used.

– If an application sends a datagram to 224.1.1.1, LINK2 is used.

Related topics

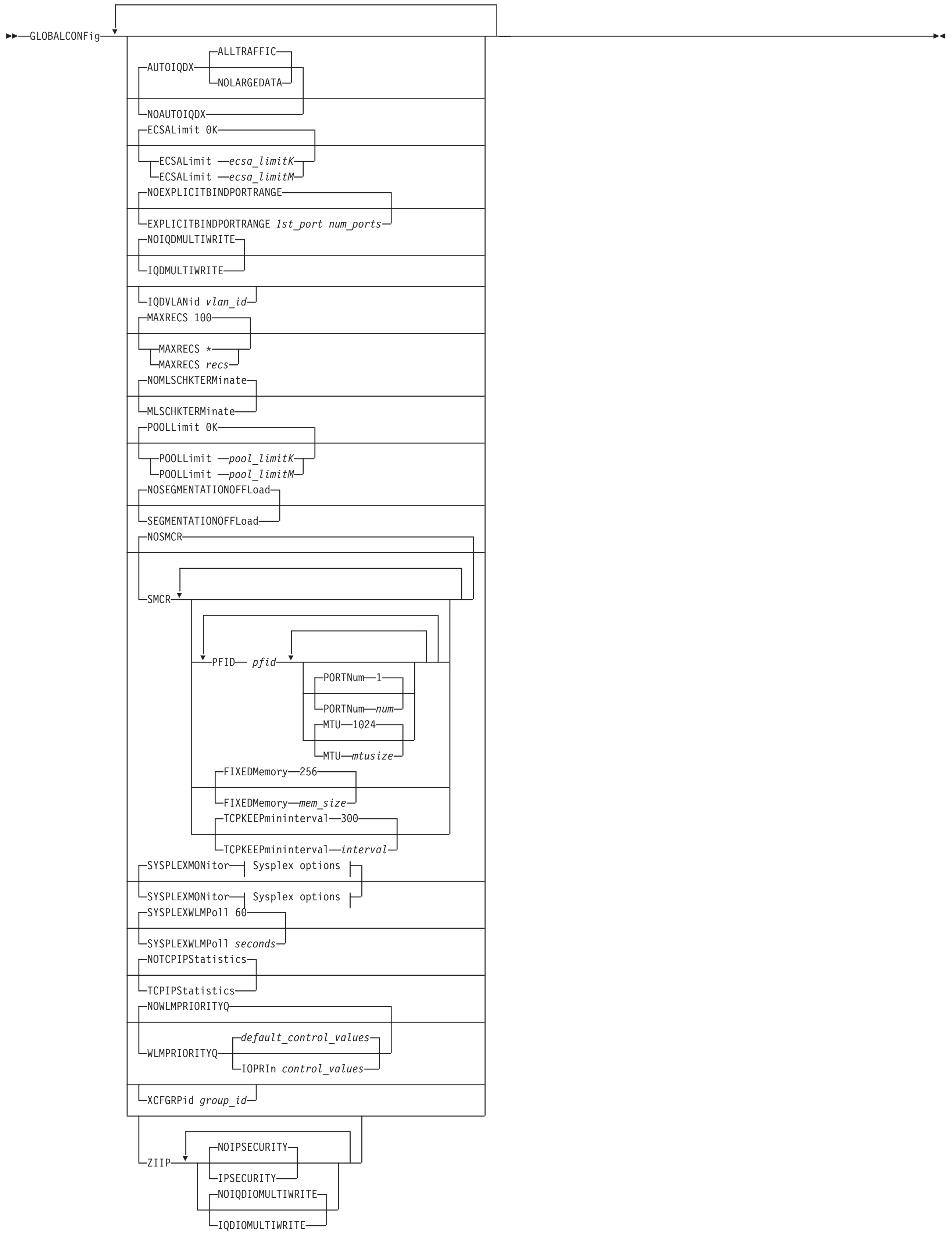
- “BEGINROUTES statement” on page 28
- “BSDROUTINGPARMS statement” on page 36
- “IPCONFIG statement” on page 190
- “RouteTable statement” on page 1152
- z/OS Communications Server: IP System Administrator's Commands

GLOBALCONFIG statement

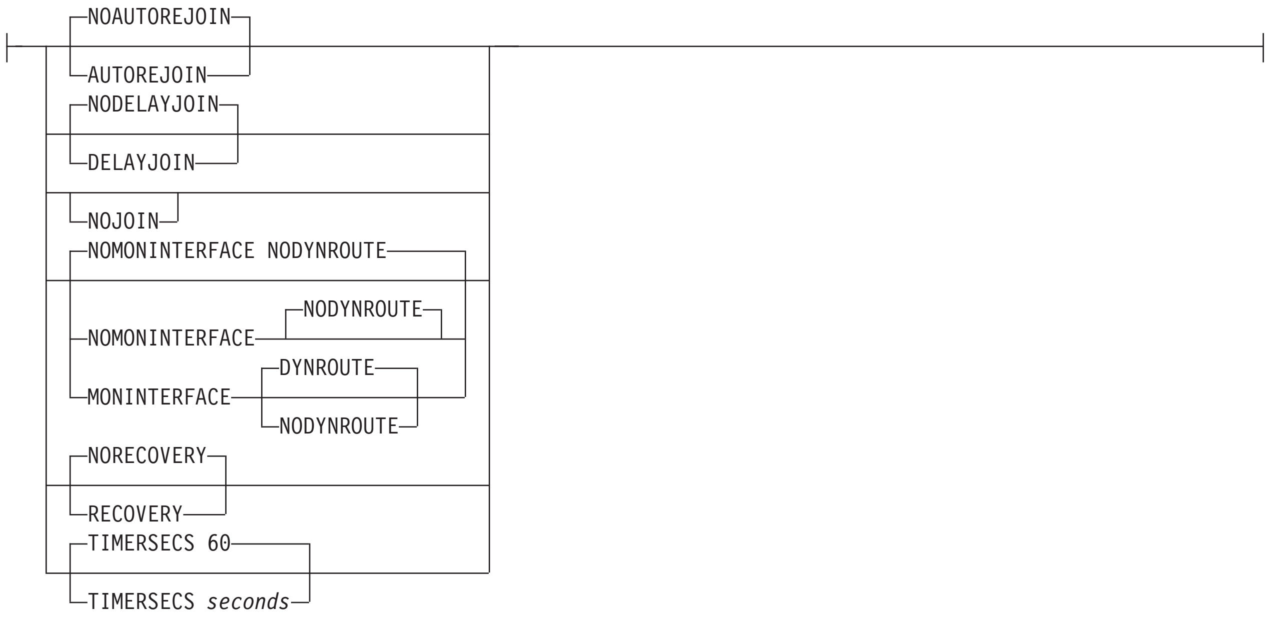
Use the GLOBALCONFIG statement to pass global configuration parameters to TCP/IP.

Syntax

Tip: Specify the parameters for this statement in any order.


GLOBALCONFig
AUTOIQDX
ALLTRAFFIC
NOLARGEDATA
NOAUTOIQDX
ECSALimit 0K
ECSALimit ecsa_limitK
ECSALimit ecsa_limitM
NOEXPLICITBINDPORTRANGE
EXPLICITBINDPORTRANGE 1st_port num_ports
NOIQDMULTIWRITE
IQDMULTIWRITE
IQDVLANid vlan_id
MAXRECS 100
MAXRECS *
MAXRECS recs
NOMLSCHKTERMinate
MLSCHKTERMinate
POOLLimit 0K
POOLLimit pool_limitK
POOLLimit pool_limitM
NOSEGMENTATIONOFFLoad
SEGMENTATIONOFFLoad
NOSMCR
SMCR
PFID pfid
PORTNum 1
PORTNum num
MTU 1024
MTU mtusize
FIXEDMemory 256
FIXEDMemory mem_size
TCPKEEPmininterval 300
TCPKEEPmininterval interval
SYSPLEXMONitor Sysplex options
SYSPLEXMONitor Sysplex options
SYSPLEXWLMPoll 60
SYSPLEXWLMPoll seconds
NOTCPIPStatistics
TCPIPStatistics
NOWLMPRIORITYQ
WLMPRIORITYQ
default_control_values
IOPRIn control_values
XCFGRPid group_id
ZIIP
NOIPSECURITY
IPSECURITY
NOIQDIOMULTIWRITE
IQDIOMULTIWRITE

Sysplex options:



Parameters

AUTOIQDX | NOAUTOIQDX

Specifies whether to use dynamic Internal Queued Direct I/O extensions (IQDX) interfaces for connectivity to the intraensemble data network.

See "Steps for modifying" on page 133 for details about changing this parameter while the TCP/IP stack is active. See z/OS Communications Server: IP Configuration Guide for information about the intraensemble data network and the dynamic IQDX function.

NOAUTOIQDX

Do not use dynamic IQDX interfaces.

AUTOIQDX

Use dynamic IQDX interfaces when an IQD CHPID has been configured with the Internal Queued Direct I/O extensions (IQDX) function. This value is the default value.

ALLTRAFFIC

Use IQDX interfaces for all eligible outbound traffic on the intraensemble data network. This value is the default value.

NOLARGEDATA

Do not use IQDX interfaces for outbound TCP socket data transmissions of length 32KB or larger. Use IQDX interfaces for all other eligible outbound traffic. See z/OS Communications Server: IP Configuration Guide for more information.

ECSALIMIT *ecsalimit* **K | M**

Specifies the maximum amount of extended common service area (ECSA) that TCP/IP can use. This limit can be expressed as a number followed by a K (which represents 1024 bytes), or a number followed by an M (which represents 1048576 bytes). If the K suffix is used, *ecsalimit* must be in the range 10240K and 2096128K inclusive or 0. If the M suffix is used, *ecsalimit* must be in the range 10M and 2047M inclusive or 0. The default is no limit, and it can be specified as 0 K or 0 M. The minimum value for ECSALIMIT and POOLLIMIT is not allowed to be set to a value if the current storage in use would be greater than or equal to 80% of that value (for example, not allowed to set it such that there is an immediate storage shortage).

ECSALIMIT ensures that TCP/IP does not overuse common storage. It is intended to improve system reliability by limiting TCP/IP's storage usage. The limit must account for peak storage usage during periods of high system activity or TCP/IP storage abends might occur. The limit does not include storage used by communications storage manager (CSM). CSM ECSA storage is managed independently of the TCP/IP ECSALIMIT. See z/OS Communications Server: SNA Network Implementation Guide for more information about CSM.

Specifying a nonzero ECSALIMIT enables warning messages EZZ4360I, EZZ4361I, and EZZ4362I to appear if a storage shortage occurs.

EXPLICITBINDPORTRANGE | NOEXPLICITBINDPORTRANGE

NOEXPLICITBINDPORTRANGE
Indicates that this stack does not participate in the allocation of ports from a pool of ports. The ports in the pool are guaranteed to be unique across the sysplex in that they are allocated to only one requestor in the sysplex at any one time, when processing an explicit bind() of a TCP socket to the IPv4 INADDR_ANY address, or to the IPv6 unspecified address (in6addr_any), and port 0.

EXPLICITBINDPORTRANGE
Indicates that this stack participates in the allocation of ports from a pool of ports guaranteed to be unique across the sysplex, when processing an explicit bind() of a TCP socket to the IPv4 INADDR_ANY address, or to the IPv6 unspecified address (in6addr_any), and port 0. This parameter also designates the range of ports that defines that pool. This parameter defines the range used by all stacks participating in EXPLICITBINDPORTRANGE port allocation processing throughout the sysplex. The most recently processed profile or OBEYFILE command that specifies EXPLICITBINDPORTRANGE defines the range for the sysplex.

Use this parameter so that you can specify distributed DVIPAs as the source IP address on DESTINATION or JOBNAME rules in a SRCIP block. See "SRCIP statement" on page 282.

1st_port
The starting port for the range of ports. The *1st_port* value is in the range 1024 - 65535. The sum of the *1st_port* value plus the *num_ports* value minus 1 cannot exceed 65535.

num_ports
The number of ports in the range. The *num_ports* value is in the range 1 - 64512. The sum of the *1st_port* value plus the *num_ports* value minus 1 cannot exceed 65535.

Guidelines:

- All TCP/IP stacks in the sysplex that participate in EXPLICITBINDPORTRANGE processing should have the same port range specified. To ensure this, specify the GLOBALCONFIG EXPLICITBINDPORTRANGE statement in a file that is specified in an INCLUDE statement in the TCP profiles data set of all the participating stacks.

- The port range defined on the EXPLICITBINDPORTRANGE parameter should not overlap any existing port reservations of any TCP/IP stacks in the sysplex. Any reserved ports that are within the EXPLICITBINDPORTRANGE range are excluded from the EXPLICITBINDPORTRANGE port pool, effectively making the pool smaller.
- The EXPLICITBINDPORTRANGE port range must be large enough to accommodate all applications in the sysplex that might issue explicit bind() calls for the IPv4 INADDR_ANY address, or for the IPv6 unspecified address (in6addr_any), and port 0.
- If additional TCP/IP stacks or systems are introduced into the sysplex, the extent of the port range defined by EXPLICITBINDPORTRANGE should be re-evaluated.
- If the size of the port range defined by the EXPLICITBINDPORTRANGE parameter is too large, there are fewer ports available for local ephemeral port allocation.
- If you specify the EXPLICITBINDPORTRANGE parameter in a sysplex that contains pre-V1R9 TCP/IP stacks, each distributor, backup, and target TCP/IP stack of a distributed SYSPLEXPORTS DVIPA that is configured as a source IP address on a SRCIP profile statement must have one of the following characteristics:
  - Run on a V1R9 or later system.
  - Use the PORTRANGE profile statement on the pre-V1R9 stacks to reserve the ports that are configured on the V1R9 or later stacks with the EXPLICITBINDPORTRANGE parameter.

  Failure to meet these characteristics can result in connection failures because unique ports assignments are no longer be assured throughout the sysplex for a SYSPLEXPORTS distributed DVIPA; the same port value could be assigned from the following pools:
  - The DVIPA-specific pool by a pre-V1R9 system
  - The EXPLICITBINDPORTRANGE pool by a V1R9 or later system

Restriction: In a common INET (CINET) environment, this parameter is accepted, but the EXPLICITBINDPORTRANGE function is supported in a limited set of conditions only. It is supported when CINET is managing one stack only on the system or when the affected application has established stack affinity. Otherwise, results can be unpredictable.

IQDMULTIWRITE | NOIQDMULTIWRITE

Specifies whether HiperSockets interfaces should use multiple write support. HiperSockets multiple write might reduce CPU usage and might provide a performance improvement for large outbound messages that are typically generated by traditional streaming workloads such as file transfer, and interactive web-based services workloads such as XML or SOAP. This parameter applies to all HiperSockets interfaces, including IUTIQDIO and IQDIOINTF6 interfaces created for Dynamic XCF.

Restriction: HiperSockets multiple write is effective only on an IBM System z10™ or later and when z/OS is not running as a guest in a z/VM® environment.

See the modifying information in this topic for details about changing this parameter while the TCP/IP stack is active. See the HiperSockets multiple write information in z/OS Communications Server: IP Configuration Guide for more information about HiperSockets multiple write support.

NOIQDMULTIWRITE
HiperSockets interfaces do not use the multiple write support. This is the default.

IQDMULTIWRITE
HiperSockets interfaces do use the multiple write support.

IQDVLANID *vlan_id*
Specifies a VLAN ID to be used when HiperSockets (iQDIO) connectivity is used for dynamic XCF support. VLAN IDs are used to partition communication across HiperSockets. Stacks on the same CPC using the same HiperSockets CHPID that use the same VLAN ID can establish communications; stacks on the same CPC using the same HiperSockets CHPID that use different VLAN IDs cannot.

The specified value, *vlan_id*, is used for both IPv4 and IPv6 DYNAMICXCF HiperSockets connectivity. This parameter is intended to be used in conjunction with the GLOBALCONFIG XCFGRPID parameter to support subplexing.

Subplexing enables TCP/IP participation in a Sysplex to be partitioned into subsets based on the XCFGRPID value. When using subplexing, TCP/IP stacks with the same XCFGRPID value should specify the same IQDVLANID value. Stacks with different XCFGRPID values should have different IQDVLANID values. If you have stacks in the default subplex (that is, stacks that do not specify an XCFGRPID value) that use the same HiperSockets CHPID as stacks within a non-default subplex (an XCFGRPID value was specified), then the stacks in the default subplex should specify an IQDVLANID value that is different from the other IQDVLANID values specified by the other non-default subplex stacks that use the same HiperSockets CHPID.

Restriction: The IQDVLANID parameter can be specified only in the initial profile.

Valid VLAN IDs are in the range 1 - 4094. For more information about VLANs and Hipersockets see z/OS Communications Server: IP Configuration Guide.

MAXRECS
Specifies the maximum number of records to be displayed by the DISPLAY TCPIP,,NETSTAT operator command. The term *records* refers to the number of entries displayed on each report. For example, for the connection-related reports, a record is a TCP connection or listener, or a UDP endpoint. This configured value is used when the MAX parameter is not explicitly specified on the command. The default value is 100. If the number of output lines exceeds the maximum number of lines for a multi-line Write to Operator (WTO), the report output is truncated. See the information about the Display TCPIP,,NETSTAT command in z/OS Communications Server: IP System Administrator's Commands for more details about the command.

*	A value of asterisk (*) specifies that all records are to be displayed.
recs	This value specifies the number of records to be displayed. The valid range is 1 - 65535.

MLSCHKTERMINATE | NOMLSCHKTERMINATE

NOMLSCHKTERMINATE
Specifies that the stack should remain active after writing an informational message when inconsistent configuration information is discovered in a multilevel-secure environment.

Informational message EZD1217I is written to the system console summarizing the number of problems found. Additional informational messages between EZD1219I and EZD1234I are written to the job log for each configuration inconsistency found.

This is the default value.

MLSCHKTERMINATE
Specifies that the stack should be terminated after writing an informational message when inconsistent configuration information is discovered in a multilevel-secure environment.

Informational message EZD1217I is written to the system console summarizing the number of problems found. Additional informational messages between EZD1219I and EZD1234I are written to the job log for each configuration inconsistency found.

POOLLIMIT *pool_limit* **K | M**
Specifies the maximum amount of authorized private storage that TCP/IP can use within the TCP/IP address space. This limit can be expressed as a number followed by a K (which represents 1024 bytes), or a number followed by an M (which represents 1048576 bytes). If the K suffix is used, *pool_limit* must be in the range 10240K and 2096128K inclusive or 0. If the M suffix is used, *pool_limit* must be in the range 10M and 2047M inclusive or 0. The default is no limit, and it can be specified as 0K or 0M. The minimum value for ECSALIMIT and POOLLIMIT is not allowed to be set to a value if the current storage in use would be greater than or equal to 80% of that value (for example, not allowed to set it such that there is an immediate storage shortage).

POOLLIMIT ensures that TCP/IP does not overuse its authorized private storage. Most systems can use the default POOLLIMIT (no limit). Systems with limited paging capacity can use POOLLIMIT to help limit TCP/IP storage usage. If the limit is used, it must account for peak storage usage during periods of high system activity or TCP/IP storage abends might occur.

POOLLIMIT can be higher than the REGION size on the TCP/IP start procedure because POOLLIMIT applies to authorized storage, whereas REGION applies to unauthorized storage. Specifying a nonzero POOLLIMIT enables warning messages EZZ4364I, EZZ4365I, and EZZ4366I to appear if a storage shortage occurs.

SEGMENTATIONOFFLOAD | NOSEGMENTATIONOFFLOAD
Specifies whether the stack should offload TCP segmentation for IPv4 packets to OSA-Express features. TCP segmentation offload support transfers the overhead of segmenting outbound data into individual TCP packets to QDIO-attached OSA-Express devices whose features that support this function. Offloading segmentation of streaming-type workloads reduces CPU use and increases throughput. This parameter is ignored for OSA-Express features that do not support segmentation offload.

Guideline: The support for specifying IPv4 segmentation offload on the GLOBALCONFIG profile statement has been deprecated. The parameters are still supported on the GLOBALCONFIG statement, but the support for specifying these parameters on the GLOBALCONFIG statement will be dropped in a future release. It is recommended to specify these parameters on the IPCONFIG profile statement instead.

Rule: The SEGMENTATIONOFFLOAD and NOSEGMENTATIONOFFLOAD parameters specified on the IPCONFIG statement override the equivalent parameters specified on the GLOBALCONFIG statement.

See the Modifying topic for information about changing this parameter while the TCP/IP stack is active. See TCP segmentation offload information in z/OS Communications Server: IP Configuration Guide for more information about TCP segmentation offload support.

NOSEGMENTATIONOFFLOAD
TCP segmentation is performed by the TCP/IP stack. This is the default.

SEGMENTATIONOFFLOAD
TCP segmentation is offloaded to the OSA-Express feature.

SMCR | NOSMCR
Specifies whether this stack uses Shared Memory Communications over Remote Direct Memory Access (RDMA), or SMC-R, for external data network communications. For more information about SMC-R, see Shared Memory Communications over Remote Direct Memory Access in z/OS Communications Server: IP Configuration Guide.

NOSMCR
Specifies that this stack should not use SMC-R for external data network communications. This is the default setting.

SMCR
Specifies that this stack should use SMC-R for external data network communications. Use this parameter to define the IBM 10GbE RoCE Express features that this stack should use for SMC-R communications. You can use this parameter to define additional operational characteristics for SMC-R communications.

If you specify the SMCR parameter without any subparameters, you get one of the following results:

- If this is the first time that you specify the SMCR parameter, no Peripheral Component Interconnect Express (PCIe) function IDs are defined and the FIXEDMEMORY and TCPKEEPMININTERVAL subparameters are set to default values.
- If you previously specified the SMCR parameter with subparameters, TCP/IP retains the knowledge of the subparameter settings, even if SMC-R processing is stopped by issuing the VARY TCPIP,,OBEYFILE command with a data set that contains a GLOBALCONFIG NOSMCR parameter. Therefore, a subsequent specification of a GLOBALCONFIG SMCR profile statement resumes SMC-R processing with the previous subparameter settings.

PFID *pfid*
Specifies the Peripheral Component Interconnect Express (PCIe) function ID (PFID) value for a 10GbE RoCE Express feature that this stack uses. A *pfid* is a 2-byte hexadecimal value
| in the range 0 - 0FFF that identifies this TCP/IP stack's
| representation of a 10GbE RoCE Express feature.

Rules:

- You must code at least one PFID subparameter for this stack to use SMC-R communications.

- You can specify a maximum of 16 PFID subparameter values on the SMCR parameter.
- The value for each PFID and PORTNUM pair must be unique.
- When the RoCE Express feature operates in a shared RoCE environment, you cannot simultaneously activate a 10GbE RoCE Express feature that uses the same PFID value from different TCP/IP stacks within the same logical partition (LPAR).

PORTNUM *num*

Specifies the 10GbE RoCE Express port number to use for a particular PFID. Configure each PFID to use only a single port. The port number can be 1 or 2; 1 is the default value.

Rules:

- If the 10GbE RoCE Express feature operates in a dedicated RoCE environment, you can activate either port 1 or port 2 but not both simultaneously for an individual PFID value. If PORTNUM 1 and PORTNUM 2 definitions for the same PFID value are created, the port that is first activated is used.
- If the 10GbE RoCE Express feature operates in a shared RoCE environment, you can use both port 1 and port 2 on an individual RNIC adapter, but the PFID value that is associated with each port must be different. You cannot simultaneously activate PORTNUM 1 and PORTNUM 2 definitions for the same PFID value.

  For example, if PFID 0013 and PFID 0014 are both defined in HCD to represent the RNIC adapter with PCHID value 0140, you can configure `PFID 0013 PORT 1 PFID 0014 PORT 2` to use both ports on the RNIC adapter. However, if you specify `PFID 0013 PORT 1 PFID 0013 PORT 2`, only the first port that is activated will be used.

MTU *mtusize*

Specifies the maximum transmission unit (MTU) value to be used for a particular PFID. The MTU value can be 1024 or 2048. The default value is 1024 and can be used for most workloads. If you set the MTU size to 2048, you must also enable jumbo frames on all switches in the network path for all peer hosts. For more information about the RoCE maximum transmission unit, see z/OS Communications Server: IP Configuration Guide.

FIXEDMEMORY *mem_size*

Specifies the maximum amount of 64-bit storage that the stack can use for the send and receive buffers that are required for SMC-R communications. The *mem_size* value is an integer in the range 30 - 9999, and represents the maximum storage in megabytes of data. The default value is 256 megabytes.

TCPKEEPMININTERVAL *interval*

This interval specifies the minimum interval that TCP keepalive packets are sent on the TCP path of an SMC-R link.

Rules:

- If a keepalive interval is also specified on the INTERVAL parameter of the TCPCONFIG statement or is set for a specific SMC-R link socket by the TCP_KEEPALIVE setsockopt() option, the largest of the three interval values is used.
- The valid range for this interval is 0-2147460 seconds, and the default is 300 seconds.
- A value of 0 disables TCP keepalive probe packets on the TCP path of an SMC-R link.
- The SO_KEEPALIVE setsockopt() option must be set for keepalive processing to be used.

Result: The TCPKEEPMININTERVAL setting has no effect on keepalive processing for the SMC-R path of an SMC-R link.

For more information about TCP keepalive processing for the TCP path and the SMC-R path of SMC-R links, see TCP keepalive in z/OS Communications Server: IP Configuration Guide.

SYSPLEXMONITOR

Specifies SYSPLEXMONITOR subparameters to configure the operation of the sysplex autonomics function. For more information about connectivity problems in a sysplex, see z/OS Communications Server: IP Configuration Guide.

If the SYSPLEXMONITOR parameter is not specified in the initial TCP/IP profile, then the sysplex autonomics function uses the default values for all SYSPLEXMONITOR subparameters. If the SYSPLEXMONITOR parameter is specified but not all subparameters are specified in the initial TCP/IP profile, then the sysplex autonomics function uses the default values for those SYSPLEXMONITOR subparameters that are not specified. For example, if SYSPLEXMONITOR is specified without RECOVERY or NORECOVERY specified in the initial profile, then the NORECOVERY action is in effect.

Rule: If you specify the GLOBALCONFIG statement in a data set associated with a VARY TCPIP,,OBEYFILE command and the SYSPLEXMONITOR parameter is specified without any subparameters, an informational message is issued and the parameter is ignored.

AUTOREJOIN | NOAUTOREJOIN

Specifies whether TCP/IP should automatically rejoin the TCP/IP sysplex group when a detected problem is relieved after the stack has left the sysplex group.

NOAUTOREJOIN

Do not rejoin the TCP/IP sysplex group when a detected problem is relieved. This is the default value.

AUTOREJOIN

When all detected problems (that caused the stack to leave the sysplex group) are relieved, the stack automatically rejoins the sysplex group and reprocesses the saved VIPADYNAMIC block configuration.

Restriction: AUTOREJOIN cannot be configured when NORECOVERY is configured (or set to the default value).

Guideline: AUTOREJOIN should be used when RECOVERY is configured to allow the stack to rejoin the sysplex group without operator intervention.

DELAYJOIN | NODELAYJOIN
Specify whether TCP/IP should delay joining or rejoining the TCP/IP sysplex group (EZBTCPCS) during stack initialization, or rejoining the sysplex group following a VARY TCPIP,,OBEYFILE command.

NODELAYJOIN
Attempt to join the TCP/IP sysplex group. When specified during stack initialization, the stack attempts to join the sysplex group. This is the default value.

DELAYJOIN
Delay joining the TCP/IP sysplex group and processing any VIPADYNAMIC block or DYNAMICXCF statements during stack initialization until OMPROUTE is started and active.

DYNROUTE | NODYNROUTE
Specifies whether TCP/IP should monitor the presence of dynamic routes over monitored network links or interfaces.

NODYNROUTE
The TCP/IP stack should not monitor the presence of dynamic routes over monitored network links or interfaces. When MONINTERFACE is not configured, this is the default value.

DYNROUTE
The TCP/IP stack should monitor the presence of dynamic routes over monitored network links or interfaces.

Tip: This level of monitoring is useful in detecting problems that OMPROUTE is having in communicating with other routing daemons on the selected network interfaces.

If no dynamic routes are present in the TCP/IP stack from that network, a specific interface attached to that network might not be active or routers attached to that network might not be active or healthy. In either case, when these conditions are detected, they provide a reasonable indication that client requests for DVIPAs or distributed DVIPAs owned by this TCP/IP stack might not reach this stack over that interface. These checks can help further qualify the state of a network interface on this TCP/IP stack. When the MONINTERFACE parameter is specified, This is the default value.

Restriction: DYNROUTE cannot be specified when NOMONINTERFACE is configured (or is the default value).

Rules:

- Specify DYNROUTE only when OMPROUTE is configured and started; otherwise, the TCP/IP stack might be forced to leave the TCP/IP sysplex group if RECOVERY is coded.
- If DYNROUTE is specified, also specify DELAYJOIN to avoid a scenario where the TCP/IP stack leaves the TCP/IP sysplex group before OMPROUTE is started.

NOJOIN
Specifies that the TCP/IP stack should not join the TCP/IP sysplex group (EZBTCPCS) during stack initialization. If this value is specified,

the TCP/IP stack does not process any VIPADYNAMIC block or DYNAMICXCF statements. Any other GLOBALCONFIG SYSPLEXMONITOR parameter settings (configured or default) are ignored, and the settings are saved in case you want the TCP/IP stack to join the sysplex group at a later time.

If you subsequently issue a VARY TCPIP,,SYSPLEX,JOINGROUP command, the NOJOIN setting is overridden and the saved GLOBALCONFIG SYSPLEXMONITOR parameter settings become active. For example, if you configure NOJOIN and DELAYJOIN, DELAYJOIN is initially ignored. If you subsequently issue a V TCPIP,,SYSPLEX,JOINGROUP command, NOJOIN is overridden, DELAYJOIN becomes active, and the stack joins the sysplex group if OMPROUTE is initialized.

Any sysplex-related definitions within the TCP/IP profile, such as VIPADYNAMIC or IPCONFIG DYNAMICXCF statements, are not processed until the TCP/IP stack joins the sysplex group.

Restriction: You can specify this parameter only in the initial profile; you cannot specify it when you issue a VARY TCPIP,,OBEYFILE command.

MONINTERFACE | NOMONINTERFACE

NOMONINTERFACE
The TCP/IP stack should not monitor the status of any network links or interfaces. This is the default.

MONINTERFACE
The TCP/IP stack should monitor the status of specified network link or interfaces. The interfaces or links being monitored are those that are configured with the MONSYSPLEX keyword on the LINK or INTERFACE statement. See "Summary of DEVICE and LINK statements" on page 47 or "Summary of INTERFACE statements" on page 141 for more information.

Guideline: This level of monitoring can further qualify the health of the TCP/IP stack by ensuring that at least one key interface is active and available. This option can be useful in environments where the dynamic XCF interface is not configured as an alternate network path for this stack (for example, where no dynamic routes are advertised over dynamic XCF interfaces and no static or replaceable static routes are defined over those interfaces).

RECOVERY | NORECOVERY
Specify the action to be taken when a sysplex problem is detected.

NORECOVERY
When a problem is detected, issue messages regarding the problem but take no further action. This is the default value.

RECOVERY
When a problem is detected, issue messages regarding the problem, leave the TCP/IP sysplex group, and delete all DVIPA resources owned by this stack. As allowed by a configuration with backup capabilities, other members of the TCP/IP sysplex automatically take over the functions of this member that was removed from the TCP/IP sysplex group.

Recovery is the preferred method of operation because other members of the TCP/IP sysplex can automatically take over the functions of a member with no actions needed by an operator. IBM Health Checker for z/OS enhancements can be used to check whether the RECOVERY parameter has been specified when the IPCONFIG DYNAMICXCF or IPCONFIG6 DYNAMICXCF parameters have been specified. For more details about IBM Health Checker for z/OS enhancements, see the IBM Health Checker for z/OS enhancements information in the z/OS Communications Server: IP Diagnosis Guide.

TIMERSECS *seconds*
Time value specified in seconds. Determines how quickly the sysplex monitor reacts to problems with needed sysplex resources. Valid values are in the range 10 - 3600 seconds. The default value is 60 seconds.

SYSPLEXWLMPOLL *seconds*
Time value specified in seconds. Determines how quickly the sysplex distributor and its target servers poll WLM for new weight values. A short time results in quicker reactions to changes in target status. Valid values are in the range is 1 - 180 seconds. The default value is 60 seconds.

TCPIPSTATISTICS | NOTCPIPSTATISTICS

NOTCPIPSTATISTICS
Indicates that the TCP/IP counter values are not to be written to the output data set designated by the CFGPRINT JCL statement.

The NOTCPIPSTATISTICS parameter is confirmed by the message:

```
EZZ0613I TCPIPSTATISTICS IS DISABLED
```

This is the default value.

TCPIPSTATISTICS
Prints the values of several TCP/IP counters to the output data set designated by the CFGPRINT JCL statement. These counters include number of TCP retransmissions and the total number of TCP segments sent from the MVS TCP/IP system. These TCP/IP statistics are written to the designated output data set only during termination of the TCP/IP address space.

The TCPIPSTATISTICS parameter is confirmed by the message:

```
EZZ0613I TCPIPSTATISTICS IS ENABLED
```

The SMFCONFIG TCPIPSTATISTICS parameter (see “SMFCONFIG statement” on page 274) serves a different purpose. It requests that SMF records of subtype 5 containing TCP/IP statistics be created. These statistics are recorded in SMF type 118 or 119, subtype 5 records.

WLMPRIORITYQ | NOWLMPRIORITYQ
Specifies whether OSA-Express QDIO write priority values should be assigned to packets associated with WorkLoad Manager service classes, and to forwarded packets. See the information about prioritizing outbound OSA-Express data using the WorkLoad Manager service class in z/OS Communications Server: IP Configuration Guide .

NOWLMPRIORITYQ
Specifies that OSA-Express QDIO write priority values should not be assigned to packets associated with WorkLoad Manager service class values or to forwarded packets. This value is the default.

WLMPRIORITYQ

Specifies that OSA-Express QDIO write priority values should be assigned to packets associated with WorkLoad Manager service class valuesand to forwarded packets.

You can assign specific OSA-Express QDIO write priority values by using the IOPRI*n* subparameters, where *n* is one or more of the priority values in the range 1 - 4. For each subparameter, you can specify a control value in the range 0 - 6, which correlates to the WLM services classes, or you can specify the keyword FWD for forwarded packets. WLM supports a service class for the SYSTEM value, but this value is always assigned the OSA-Express QDIO write priority 1 and its assignment cannot be configured; therefore, a control value is not assigned for the SYSTEM WLM service class.

You can use the default assignment by specifying the WLMPRIORITYQ parameter without any IOPRI*n* subparameters. See the description of the *default_control_values* variable in this topic to understand the default assignment.

control_values

Control values are used to represent the WLM service classes and forwarded packets. Valid control values are the digits 0 - 6, which represent WLM service classes, or the keyword FWD, which represents forwarded packets. Table 4 identifies the control value, the type of packet that it represents, and the default QDIO priority assigned to the packet:

Table 4. WLM Service Class Importance Levels

Control value	Type of packet	Default QDIO priority
0	System-defined service class (SYSSTC) used for high-priority started tasks	1
1	User-defined service classes with importance level 1	2
2	User-defined service classes with importance level 2	3
3	User-defined service classes with importance level 3	3
4	User-defined service classes with importance level 4	4
5	User-defined service classes with importance level 5	4
6	User-defined service classes associated with a discretionary goal	4
FWD	Forwarded packets	4

default_control_values

When the WLMPRIORITYQ parameter is specified without any IOPRIn subparameters, then the OSA-Express QDIO write priority values are assigned as shown Table 4.

IOPRI*n control_values*

Use the IOPRIn subparameters to correlate control values with specific OSA-Express QDIO write priority values. You can use one or more of the following subparameter keywords:

- IOPRI1
- IOPRI2
- IOPRI3
- IOPRI4

Each subparameter keyword corresponds to one of the four QDIO write priority values, 1 through 4. Each subparameter can be specified once on a GLOBALCONFIG statement.

control_values
Indicates the type of packet to which the QDIO write priority value should be assigned. Valid values are:

Digits 0 - 6
Causes the QDIO write priority value that is specified by the IOPRI*n* subparameter to be assigned to packets associated with the WLM service classes represented by the control value.

FWD This keyword causes the QDIO write priority value indicated by the IOPRIn subparameter to be assigned to forwarded packets.

Rules:

- IOPRIn must be followed by one or more priority level releases.
- You can specify more than one control value for an IOPRIn subparameter. Each control value must be separated by at least one blank.
- A specific control value can be specified only once in the set of IOPRIn subparameters on a GLOBALCONFIG statement.
- If any control value is not explicitly specified on an IOPRI*n* subparameter, then the associated packets are assigned a default QDIO write priority 4.

In the following example, QDIO priority 1 is assigned to packets associated with control values 0 and 1, QDIO priority 2 is assigned to packets associated with control value 2 and to forwarded packets, QDIO priority 3 is assigned to packets associated with control values 3 and 4, and QDIO priority 4 is assigned to packets associated with control values 5 and 6.

```
WLMPRIORITYQ  IOPRI1 0 1
              IOPRI2 2 FWD
              IOPRI3 3 4
              IOPRI4 5 6
```

XCFGRPID *group_id*
This parameter is needed only if you want subplexing. If specified, the value provides a 2-digit suffix that is used in generating the XCF group name that the TCP/IP stack joins. Valid values are in the range 2 - 31. The group name is EZBT*vvtt*, where the *vv* value is the VTAM XCF group ID suffix (specified with the XCFGRPID VTAM start option) and the *tt* value is the *group_id* value supplied on this parameter, used as a 2-digit value converted to character format. If no VTAM XCF group ID suffix was specified, the group name is EZBTCP*tt.* If no VTAM XCF group ID suffix and no TCP XCF group ID suffix is specified, the group name is EZBTCPCS.

These characters are also used as a suffix for the EZBDVIPA and EZBEPORT structure names, in the form EZBDVIPA*vvtt* and EZBEPORT*vvtt*. If no VTAM XCF group ID suffix was specified, the structure names are EZBDVIPA01*tt* and EZBEPORT01*tt*.

If XCFGRPID is not specified, the XCF group name is EZBTvvCS and the structure names are EZBDVIPA*vv* and EZBEPORT*vv*. If no VTAM XCF group id suffix was specified, the group name is EZBTCPCS and the structure names are EZBDVIPA and EZBEPORT.

Restriction: XCFGRPID can be specified only in the initial profile.

This allows multiple TCP/IP stacks to join separate Sysplex groups and access separate Coupling Facility structures, isolating sets of TCP/IP stacks into subplexes with XCF communication only with other TCP/IP stacks within the same subplex.

If HiperSockets is supported on this system, the IQDVLANID parameter, on the GLOBALCONFIG statement, must be specified if XCFGRPID is specified. Stacks on the same CPC using the same HiperSockets CHPID that specify the same XCFGRPID value must specify the same IQDVLANID value.

Stacks on the same CPC using the same HiperSockets CHPID specifying different XCFGRPID values must specify different IQDVLANID values. This allows partitioning of connectivity across the Sysplex to include partitioning of connectivity across HiperSockets.

Creating TCP/IP and VTAM subplexes can add some complexity to your VTAM and TCP/IP configurations and requires careful planning. Before setting this parameter you should review the information about setting up a subplex in the z/OS Communications Server: IP Configuration Guide.

ZIIP

Specifies subparameters that control whether TCP/IP displaces CPU cycles onto a System z9® Integrated Information Processor (zIIP). You must specify at least one subparameter. If the ZIIP parameter is specified with no subparameters, an informational message is issued and the parameter is ignored.

IPSECURITY | NOIPSECURITY

Specifies whether TCP/IP should displace CPU cycles for IPSec workload to a zIIP. For more information about this function, see the Additional IPSec assist using z9 Integrated Information Processor (zIIP IP security) topic in z/OS Communications Server: IP Configuration Guide.

NOIPSECURITY

Do not displace CPU cycles for IPSec workload to a zIIP. This is the default value.

IPSECURITY

When possible, displace CPU cycles for IPSec workload to a zIIP. Workload Manager (WLM) definitions should be examined and possible changes made before this option is used. See the more detailed description in the additional IPSec Assist by way of z9 Integrated Information Processor (zIIP IPSECURITY) topic in z/OS Communications Server: IP Configuration Guide.

NOIQDIOMULTIWRITE | IQDIOMULTIWRITE

Specifies whether TCP/IP should displace CPU cycles for large outbound TCP messages that are typically created by traditional streaming work loads such as file transfer, and interactive web-based service workloads such as XML or SOAP. The TCP/IP outbound message must be at 32KB in length before the write processing is off-loaded to an available zIIP specialty engine. For more information about this function, see the information about additional IPSec Assist by way of z9 Integrated Information Processor (zIIP IPSECURITY) in z/OS Communications Server: IP Configuration Guide.

NOIQDIOMULTIWRITE
: Do not displace CPU cycles for the writing of large TCP outbound messages to a zIIP. This is the default value.

IQDIOMULTIWRITE
: When possible, displace CPU cycles for the writing of large TCP outbound messages to a zIIP.

Rules:

- You cannot specify IQDIOMULTIWRITE as a ZIIP parameter when GLOBALCONFIG IQDMULTIWRITE is not configured. When GLOBALCONFIG IQDMULTIWRITE is not configured, HiperSockets interfaces do not use the multiple write support.
- Only large TCP outbound messages (32KB and larger) are processed on the zIIP specialty engine.
- The TCP message must be originating from this node. Routed TCP messages are not eligible for zIIP assistance.

Tip: These ZIIP parameters apply to pre-defined HiperSockets interfaces, as well as HiperSockets interfaces that are created and used by dynamic XCF definitions.

Steps for modifying

To modify parameters for the GLOBALCONFIG statement, you must respecify the statement with the new parameters.

The following list describes how to modify individual parameters:

AUTOIQDX and NOAUTOIQDX
: If you use the VARY TCPIP,,OBEYFILE command to change this parameter from AUTOIQDX to NOAUTOIQDX, no new dynamic IQDX interfaces will be activated. All active dynamic IQDX interfaces will remain active and available for use. To stop existing interfaces, you must issue a V TCPIP,,STOP command for each active IQDX interface.

  If you use the VARY TCPIP,,OBEYFILE command to change this parameter from NOAUTOIQDX to AUTOIQDX, active OSX interfaces are not affected, but the stack will attempt to activate a dynamic IQDX interface on any subsequent OSX activations.

EXPLICITBINDPORTRANGE and NOEXPLICITBINDPORTRANGE
: If you specified the EXPLICITBINDPORTRANGE parameter and then you change to the NOEXPLICITBINDRANGE parameter, then the stack stops allocating more ports from the EXPLICITBINDPORTRANGE pool. However, the existing active range for the EXPLICITBINDPORTRANGE pool in the coupling facility is unaffected unless you are changing the parameter on the last stack in the sysplex using this function.

  If you specified the NOEXPLICITBINDPORTRANGE parameter and then you change to the EXPLICITBINDPORTRANGE parameter, then a range of ports used for the EXPLICITBINDPORTRANGE pool is set. The stack uses ports from that pool for explicit bind() requests to the IPv4 INADDR_ANY address, or to the IPv6 unspecified address (in6addr_any), and port 0. If the range specified on the EXPLICITBINDPORTRANGE parameter is different from the currently active range for the EXPLICITBINDPORTRANGE pool in the coupling facility, the new range replaces that value.

Changing the starting port (*1st_port*), the number of ports (*num_ports*), or both for the EXPLICITBINDPORTRANGE parameter changes the port numbers in the pool of ports that is guaranteed to be unique across the sysplex for future port allocation

Guidelines:

- Changing the range specified on the EXPLICITBINDPORTRANGE parameter of the GLOBALCONFIG statement affects every stack in the sysplex that has configured a GLOBALCONFIG EXPLICITBINDPORTRANGE value. Future port allocations for all such stacks use the new port range.
- Ports in the EXPLICITBINDPORTRANGE range are usually assigned to a stack in blocks of 64 ports. When expanding the range, use multiples of 64 multiplied by the number of stacks that use a GLOBALCONFIG EXPLICITBINDPORTRANGE configuration.

IQDMULTIWRITE and NOIQDMULTIWRITE
If this parameter is changed with the VARY TCPIP,,OBEYFILE command, the new value does not take effect for any active HiperSockets (iQDIO) interfaces. For a change in this parameter to take effect for an active iQDIO interface, you must stop and restart both the IPv4 and IPv6 interface for the change to be effective.

IQDVLANID
If the IQDVLANID parameter was previously specified and you modify that value, then you must stop and restart the TCP/IP stack for the change to take effect.

MLSCHKTERMINATE
You cannot change the MLSCHKTERMINATE parameter to the NOMLSCHKTERMINATE parameter when the RACF® option MLSTABLE is on and the RACF option MLQUIET is off. You can always change the NOMLSCHKTERMINATE parameter to the MLSCHKTERMINATE parameter, but this change is ignored if the value is specified in the data set of a VARY TCPIP,,OBEYFILE command and consistency errors are detected at the same time.

SEGMENTATIONOFFLOAD and NOSEGMENTATIONOFFLOAD:
If this parameter is changed with the VARY TCPIP,,OBEYFILE command, the new value does not take effect for any active OSA-Express QDIO interfaces. For a change in these parameters to take effect, all the OSA-Express QDIO interfaces that support TCP segmentation offload must be stopped and restarted.

SMCR and NOSMCR

- If SMCR support is not enabled, you can specify the SMCR parameter in a VARY TCPIP,,OBEYFILE command data set to activate the support.

  Result: TCP/IP retains knowledge of the last set of SMCR subparameter values that are specified on the GLOBALCONFIG statement, even if GLOBALCONFIG NOSMCR was specified subsequently. If you issue a VARY TCPIP,,OBEYFILE command with GLOBALCONFIG SMCR specified, TCP/IP uses the saved last set of SMCR subparameters, unless new values for the subparameters are coded on the GLOBALCONFIG SMCR statement. This allows you to temporarily stop SMC-R processing by issuing a VARY TCPIP,,OBEYFILE command with GLOBALCONFIG NOSMCR specified. Then you can resume SMC-R processing with the previous subparameter settings by issuing a second VARY TCPIP,,OBEYFILE command with just GLOBALCONFIG SMCR specified.

- If SMCR support is enabled, you can specify the NOSMCR parameter in a VARY TCPIP,,OBEYFILE command data set to deactivate the support.
  - No new TCP connections that use SMC-R processing will be established.
  - Existing TCP connections that use SMC-R will continue to use SMC-R processing.
- You cannot change the SMCR PFID parameter values that are currently configured when the associated 10GbE RoCE Express interfaces are active. To change the SMCR PFID parameter values that are currently configured, you must perform the following steps in order:
  1. Stop the associated 10GbE RoCE Express interfaces.
  2. Issue the VARY TCPIP,,OBEYFILE command with the new PFID values that are coded in the command data set. The new PFID values replace the existing PFID values.
- To add PFID values when you have one or more PFID values coded, you must specify the existing PFID values and the additional PFID values on the SMCR parameter in the VARY TCPIP,,OBEYFILE command data set. Existing PFID values and any existing 10GbE RoCE Express interfaces are not affected.

SYSPLEXMONITOR

AUTOREJOIN and NOAUTOREJOIN

If you change NOAUTOREJOIN to AUTOREJOIN after the stack has left the sysplex and before the problem that caused it to leave has been relieved, the stack automatically rejoins the sysplex group when the problem is relieved. However, if you change NOAUTOREJOIN to AUTOREJOIN after the problem that caused the stack to leave the group has been relieved, you must issue a VARY TCPIP,,SYSPLEX,JOINGROUP command to cause the stack to rejoin the sysplex.

DELAYJOIN and NODELAYJOIN

Changing from DELAYJOIN to NODELAYJOIN while the TCP/IP stack is in the process of delaying joining the sysplex group because OMPROUTE is not active causes the TCP/IP stack to immediately join the sysplex group.

Changing from NODELAYJOIN to DELAYJOIN has no immediate effect until the TCP/IP stack leaves the sysplex group and then attempts to rejoin while OMPROUTE is not active.

SYSPLEXWLMPOLL

You can change the polling rate for WLM values while the TCP/IP stack is active. In order for the change to be effective, you should change the polling rate on all stacks that participate in sysplex distribution (all active distributing stacks, any backup stacks that might take over distribution, and all target stacks).

WLMPRIORITYQ

If you specify WLMPRIORITYQ with the VARY TCPIP,,OBEYFILE command, the IOPRI*n* values are changed to the values specified for the *default_control_values* variable. The new values take effect immediately for all workloads influenced by this function.

WLMPRIORITYQ IOPRIn *control_values*

If you specify this parameter with the VARY TCPIP,,OBEYFILE command, and you do not specify all the control values, the QDIO priority 4 is assigned to packets associated with all control values omitted. The new values immediately take effect for all workloads influenced by this function.

Rule: You cannot modify individual IOPRI*n* control values. If you attempt to modify IOPRI*n* control values, but you specify only those control values that you want to modify, then the QDIO priority 4 is assigned to packets that are associated with any control values that you omitted.

XCFGRPID

For a change in this parameter to take effect, you must stop and restart the TCP/IP stack.

Examples

This example shows the use of the SYSPLEXMONITOR parameter on the GLOBALCONFIG statement that enables many of the sysplex autonomics functions:

```
GLOBALCONFIG SYSPLEXMONITOR AUTOREJOIN DELAYJOIN MONINTERFACE DYNROUTE RECOVERY
```

The following example shows the use of the EXPLICITBINDPORTRANGE parameter to define 1024 ports in the range 5000 - 6023. The ports are used for explicit binds to the IPv4 INADDR_ANY address, or to the IPv6 unspecified address (in6addr_any), and port 0:

```
GLOBALCONFIG EXPLICITBINDPORTRANGE 5000 1024
```

The following example shows the use of the SMCR parameter to define two 10GbE RoCE Express features that use PFID values 0018 and 0019 and port numbers 1 and 2, and to limit the stack to 500 megabytes of 64-bit storage for SMC-R communications.

```
GLOBALCONFIG SMCR PFID 0018 PORTNUM 1 PFID 0019 PORTNUM 2 FIXEDMEMORY 500
```

Related topics

- "SMFCONFIG statement" on page 274
- For more information about TCP/IP networking in a multilevel-secure environment, see the security information in z/OS Communications Server: IP Configuration Guide.

HOME statement

Use the HOME statement to provide the list of home IPv4 addresses and associated link names.

Restriction:

- The HOME statement applies only to IPv4 interfaces that are defined with DEVICE and LINK statements. Specify the home IP address on the INTERFACE statement for the following interfaces:

– An IPv6 interface.
– An IPv4 interface that is defined by using the INTERFACE statement for QDIO Ethernet, HiperSockets, or static VIPA.

See “Summary of INTERFACE statements” on page 141 for more information.

Syntax

Rule: Specify the parameters in the order shown here.

```
             ┌──────────────────────────┐
             ▼                          │
►►──HOME─────┴──────────────────────────┴─────────────────────────────►◄
              └─internet_addr──link_name─┘
```

Parameters

internet_addr
The IP address valid for this host. The IP address can be associated with a physical or VIPA link.

Requirement: The IP address must be specified in dotted decimal form.

link_name
The name of the link defined in a previous LINK statement (or the reserved name LOOPBACK) that is associated with the home address.

Steps for modifying

To modify the HOME statement, use a VARY TCPIP,,OBEYFILE command with a data set that defines a new HOME statement.

Rules:
- If you use the HOME statement to change the IP addresses of any links, you should stop and restart the affected devices. Furthermore, if the OSPF dynamic routing protocol is being used over an affected interface, you should wait between stopping and restarting the device to enable the OSPF protocol to fully propagate the deletion of the old IP address. The duration of this wait should be at least three times the dead router interval configured for the interface.
- The first HOME statement of each configuration data set that is set replaces the existing HOME list with the new list. Subsequent HOME statements in the same data set add entries to the list; however, dynamically defined HOME list entries created by XCF dynamics, by a VIPADEFINE statement, or by an application binding to an IP address in a currently valid VIPARANGE statement are not deleted by a new HOME statement. You can display dynamically created HOME list entries with the Netstat HOME/-h command. A dynamic XCF HOME list entry has the link name EZASAMEMVS or begins with EZAXCF. A dynamic VIPA HOME list entry has a link name that begins with VIPL, followed by the hexadecimal value of its IP address.
- If the first HOME statement of a profile contains no entries, then all IP addresses that were specified in a HOME statement from a previous profile are removed from the HOME list.
- If you change the IP address of a link that was used by previously specified default routes and you want to maintain those default routes, you must include your GATEWAY or BEGINROUTES statements in the VARY TCPIP,,OBEYFILE

command data set that contains the new HOME list. If you do not include your GATEWAY or BEGINROUTES statements, the static routes using that link are deleted.

- If you had previously specified the PRIMARYINTERFACE statement and want to preserve the primary interface that was previously specified, you must include your PRIMARYINTERFACE statement in the VARY TCPIP,,OBEYFILE command data set that contains the new HOME list. If you do not include the original PRIMARYINTERFACE statement, the primary interface is reset to the first entry in the new HOME list.

Usage notes

- Only one home address can be associated with a link. If the same link is specified in more than one HOME list entry, only the home address in the last entry is associated with the link. The only exception to this is the LOOPBACK link.
- The default LOOPBACK address of 127.0.0.1 is internally defined by the TCP/IP stack. If you try to define this LOOPBACK address, it is flagged as a duplicate entry. You can use a *link_name* value of LOOPBACK in the HOME list to define additional LOOPBACK addresses. No DEVICE or LINK statement is needed for LOOPBACK, and it cannot be started or stopped (LOOPBACK is always active).
- IP addresses from 127.0.0.128 through 127.0.0.255 are reserved for IBM use and cannot be coded on the HOME statement as the IP address of any link; this includes LOOPBACK addresses.
- To improve server application performance, use a non-loopback home address instead of a loopback address. This can result in improved throughput for applications that reside on the same MVS system and communicate with one another on the same TCP/IP stack.
- A HOME address used by an ATM LINK referencing an ATMLIS should be within the logical IP subnetwork defined by the LIS subnet_value and subnet_mask. If it is not within the subnetwork, the link cannot be used for sending or receiving any ATM SVC traffic.
- If a default local address is not specified using the PRIMARYINTERFACE statement, the first address in the HOME list is used as the default local address. This default local address is the value obtained by the GETHOSTID() function.
- If an outgoing packet has the limited broadcast address (255.255.255.255) as its destination address and the source address is not specified by the sender, the default local address (see previous bullet) is used as the source address as long as it is associated with a link (other than LOOPBACK) that supports broadcast. If the link associated with the default local address is LOOPBACK or it does not support broadcast, the first address in the HOME list that is associated with a link (other than LOOPBACK) that supports broadcast is used as the source address.
- When an incorrect HOME entry is encountered, all entries following that entry on that HOME statement are ignored. Subsequent HOME statements are processed.
- When defining static VIPA addresses, observe the following rules:
  - Code a primary VIPA address first in the HOME list or on the PRIMARYINTERFACE statement to serve as the default local address for use by the GETHOSTID() function.

    A static VIPA address must be a unique host address in the network and not be a duplicate of any physical IP address in the network.

If using the RIP routing protocol and host route broadcasting is not supported by adjacent routers (that is, adjacent routers are unable to learn host routes), the following restrictions for VIPA addresses must be applied in order to benefit from fault tolerance support:

- If you use subnetting and VIPA addresses are in the same network as the physical IP addresses, the subnetwork portion of any VIPA addresses must not be the subnetwork portion of any physical IP addresses in the network. In this case, assign a new subnetwork for the VIPA address.

  If subnetting is not used on any physical interface, the network portion of any VIPA address must not be the network portion of any physical IP address in the network. In this case, assign a new network for the VIPA address, preferably a class C network address.

If using the RIP routing protocol and host route broadcasting is supported by adjacent routers (that is, adjacent routers are able to learn host routes), the network or subnetwork portions of VIPA addresses can be the same across multiple z/OS TCP/IP stacks in the network. See Chapter 11, "OMPROUTE," on page 483 for more information.

While a VIPA address can be assigned to each TCP/IP stack in one z/OS image, you should define an internal point-to-point link (for example, CTC) between the stacks. This ensures that the VIPA address in one z/OS TCP/IP stack attached to a failing adapter/controller (for example, 3172) can be reached by way of another z/OS TCP/IP stack channel-attached to the same controller through another adapter or to another controller across the point-to-point link.

For information about what routing protocols to use to achieve nondisruptive TCP-connection fault tolerance, see the VIPA information in z/OS Communications Server: IP Configuration Guide.

If you are using a name server to resolve host names by way of UDP and any of the related resolver configuration files have only one name server address coded that specifies a VIPA address, the host the name server is running on must be configured to use the SOURCEVIPA option.

– In general, the static VIPA addresses can be coded in any order in the HOME list; however, if you specify the SOURCEVIPA option on the IPCONFIG statement, the order of the VIPA addresses is important in terms of how source IP addresses are used for outbound datagrams originating at the host. In this case, the following rules apply:
  - In the HOME list, the static VIPA address that precedes a physical IP address is used as the source IP address if not overridden by another method of source address selection. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guidefor the hierarchy of ways that the source IP address of an outbound packet is determined.
  - If static VIPA addresses are coded after all of the physical IP addresses, no VIPA addresses are used as the source IP address.
– More than one VIPA address can be defined in one network or subnetwork.
– You can use the VIPA address as the primary or only destination for the name of a z/OS server on the domain name server. A workstation on the network would use the z/OS server name (translated into the VIPA address) to access applications on the z/OS server.
– If you use DEVICE and LINK statements to define an IPv4 interface and you want to designate a static VIPA as the source VIPA for that interface, use DEVICE and LINK statements to define the static VIPA.

– If you use the INTERFACE statement to define an IPv4 interface and you want to designate a static VIPA as the source VIPA for that interface, take the following steps:
  - Use the INTERFACE VIRTUAL statement to define the static VIPA that will be used as the source VIPA.
  - Specify the SOURCEVIPAINTERFACE parameter on the INTERFACE statement for the IPv4 interface.

Examples

This example shows a HOME statement that defines the IP addresses of each link in the host.

```
HOME
   151.4.1.2      TR2
   192.1.1.1      VIPA1
   130.50.75.1    TR1
   193.5.2.1      ETH1
   192.2.1.1      VIPA2
   9.67.43.110    FDDI1
   193.7.2.1      SNA1
```

VIPA1 and VIPA2 are examples of static VIPA links associated with static VIPA addresses, the other values are examples of physical links associated with physical IP addresses. If you specify SOURCEVIPA on the IPCONFIG statement, VIPA1 serves as the VIPA address for TR1 and ETH1, and VIPA2 for links FDDI1 and SNA1. Because there is no VIPA definition preceding TR2 in the HOME list, it is not affected by SOURCEVIPA. The VIPA addresses are used in the outbound IP datagrams. For more information, see "IPCONFIG statement" on page 190.

The following example shows the definition of an additional LOOPBACK address:

```
HOME  9.67.113.105   CTCD00   ; CTC IP address for this system
      127.0.0.2      LOOPBACK ; additional LOOPBACK address
```

If using the SOURCEVIPA option for the outbound datagrams originating at a z/OS TCP/IP stack, see the following example for details:

```
HOME
  172.2.1.1     VIPA1  ; <-- Source for ETH1 and TR1
  151.2.3.1     ETH1
  151.4.1.1     TR1
  172.2.1.2     VIPA2  ; <-- Source for ETH2 and TR2
  151.2.3.2     ETH2
  151.4.1.2     TR2
```

Select a VIPA address in the HOME statement to provide as the local address. The address that closely precedes a physical IP address is used as the local address. For example:

Optionally, additional VIPA addresses can be defined to associate a group of interfaces and serve as local addresses. In this example, VIPA1 is associated with ETH1 and TR1, and VIPA2 is associated with ETH2 and TR2.

If an outbound datagram is not to contain a SOURCEVIPA address for a particular interface (that is, use a physical IP address), then use the following example:

```
HOME
  151.4.1.1     TR1    ; <-- No SOURCEVIPA for outbound on TR1
  172.2.1.1     VIPA   ; <-- Source for ETH1 and TR2
  151.2.3.1     ETH1
  151.4.1.2     TR2
```

If traffic over an interface should not use a source VIPA address, put the HOME entry for that interface before all VIPA addresses in the HOME list.

Related topics

- "BEGINROUTES statement" on page 28
- "BSDROUTINGPARMS statement" on page 36
- "GATEWAY statement" on page 109
- "PRIMARYINTERFACE statement" on page 270
- See the SOURCEVIPA information in "IPCONFIG statement" on page 190.

INCLUDE statement

This statement causes profile statements from the named data set to be included at the point that the INCLUDE statement is encountered. In general, a profile statement must begin and end within the same data set. For example, the statement beginning with BSDROUTINGPARMS and ending with ENDBSDROUTINGPARMS must be contained within the same data set.

Syntax

```
►►──INCLude──data_set_name──────────────────────────────────────────────►◄
```

Parameters

data_set_name
A fully qualified data set name that identifies a sequential file. The sequential file can be a sequential data set or a PDS with the member name. It cannot be a z/OS UNIX file.

Steps for modifying

Modification is not applicable to this statement.

Summary of INTERFACE statements

Use the INTERFACE statement to define an IPv6 interface. You can also use the INTERFACE statement to define an IPv4 interface for OSA-Express QDIO Ethernet. You can use the INTERFACE statement to define an IPv4 interface for OSA-Express QDIO Ethernet, HiperSockets, or static VIPA, or to define an IPv6 interface

See "Summary of DEVICE and LINK statements" on page 47 for IPv4 support for other interface types.

Table 5 on page 142 summarizes information about the IPv4 network interface types supported by TCP/IP with the INTERFACE statement.

Table 5. IPv4 network interface types supported by TCP/IP

Interface type	Connectivity	ID in TCP/IP profile	MTU	TRLE definition	DYNAMICXCF support
IPAQENET	LAN by way of OSA-Express in QDIO mode (Gigabit Ethernet, 10G, or Fast Ethernet), 1000BASE-T Ethernet	OSA-Express port name	8992 (1492 for Fast Ethernet)	Required	No
IPAQIDIO	Another TCP/IP within same CPC	CHPID	57344 **Note:** Based on frame size configured in HCD.	Reserved name	Yes

Table 6 summarizes information about the IPv6 network interface types supported by TCP/IP.

Table 6. IPv6 network interface types supported by TCP/IP

Interface type	Connectivity	ID in TCP/IP profile	MTU	TRLE definition	DYNAMICXCF support
IPAQENET6	LAN by way of OSA-Express in QDIO mode (Gigabit Ethernet, 10G, 1000BASE-T, or Fast Ethernet)	OSA-Express port name	9000 (1500 for Fast Ethernet)	Required	No
IPAQIDIO6	Another TCP/IP within same CPC	CHPID	57344 (2)	Reserved name	Yes
MPCPTP6 for ESCON	ESCON to another z/OS Communications Server image, running IPv6 at z/OS level V1R5 or later.	TRLE Name	59392 (1)	Manual definition required	No
MPCPTP6 for XCF	Coupling Facility or ESCON channel to another z/OS Sysplex member running IPv6 at z/OS level V1R5 or later.	CP name of target VTAM	55296	Automatically generated by VTAM	Yes
MPCPTP6 (for IUTSAMEH)	Simulated IPv6 channel to another TCP/IP (or to VTAM, for Enterprise Extender) on same z/OS image.	IUTSAMEH	65535	Automatically generated by VTAM	Yes
Note: 1. Based on MAXBFRU value in the TRLE definition. 2. Based on frame size configured in HCD.					

The stack supports one IPv4 home address per interface. The stack does not impose any limit on the number of IPv6 home addresses allowed for a given interface.

Statement dependency

The INTERFACE statement for an IPv6 interface type is rejected unless the stack is enabled for IPv6. To enable the stack for IPv6, see z/OS Communications Server: IPv6 Network and Application Design Guide for information about defining TCP/IP as a UNIX System Services physical file system (PFS).

Modifying INTERFACE statements

To modify most INTERFACE statement parameters, you must first delete and redefine the INTERFACE statement.

However, the following INTERFACE statement parameters are dynamically modifiable:
- MONSYSPLEX
- NOMONSYSPLEX

To modify these parameters on an INTERFACE statement, use a VARY TCPIP,,OBEYFILE command with a data set that contains an INTERFACE statement for an existing interface name which has new values for these parameters.

Restrictions:
- Any changes to non-modifiable parameters are ignored
- If any modifiable parameters are not specified, prior values remain in effect for these parameters

Restrictions on IPv6 addresses configured in the TCP/IP profile

The following IPv6 addresses are not accepted for *ipaddr_spec*:
- Link local IP addresses
- Multicast IP addresses
- IPv4-mapped IPv6 addresses
- Addresses with the reserved prefix ::/96
- Default loopback address (::1)
- Unspecified address (::)
- Any address where bit 6 (the universal/local flag - 'U' bit) or bit 7 (the group/individual flag - 'G' bit) of the Interface ID portion is nonzero.

  The Interface ID portion is the lower 64 bits of the address. The Interface ID bit positions are numbered 0 - 63. This is shown in the following code example:
```
|               1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|xxxxxxUGxxxxxxxx|xxxxxxxxxxxxxxxx|xxxxxxxxxxxxxxxx|xxxxxxxxxxxxxxxx|
+----------------+----------------+----------------+----------------+
```
- ISATAP address ('00005EFE'x in bits 0 - 31 of the Interface ID portion of the address).

- Reserved Anycast address (Non-multicast format prefix 001 - - 111 and 'FCFFFFFFFFFFFFF8'x in bits 0 - 56 of the Interface ID portion of the address. The format prefix is the bit string consisting of the first 3 bits of the address.)
- A site-local address that has any value other than 0 in bits 10 - 47 of the address. (A site-local address has 1111111011 in bits 0 - 9 of the address.)

  Guideline: Site-local addresses were designed to use private address prefixes that could be used within a site without the need for a global prefix. Until recently, the full negative impacts of site-local addresses in the Internet were not fully understood. Because of problems in the use and deployment of addresses constructed using a site-local prefix, the IETF has deprecated the special treatment given to the site-local prefix. An IPv6 address constructed using a site-local prefix is now being treated as a global unicast address. The site-local prefix might be reassigned for other use by future IETF standards action.

  You should not use site-local unicast addresses. Instead of site-local addresses, you should use global unicast addresses.

Steps for modifying INTERFACE statements

This topic describes the steps for modifying the INTERFACE statement.

Procedure

Perform the following steps to modify all other parameters (other than MONOSYSPLEX and NOMONOSYSPLEX) on an INTERFACE statement:

1. Stop the interface.
2. Use a VARY TCPIP,,OBEYFILE command with a data set that contains: INTERFACE *interface_name* DELETE *statement*
3. Use a VARY TCPIP,,OBEYFILE command with a data set that contains the changed INTERFACE statement.

   Rule: The data set that you use on the VARY TCPIP,,OBEYFILE command in this step should be different from the data that you used in Step 2. Do not attempt to delete and redefine an interface in the same OBEYFILE data set.
4. Start the interface.

Results

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands.

To modify IPADDR values, use INTERFACE ADDADDR and INTERFACE DELADDR.

To modify TEMPPREFIX for an IPAQENET6 interface, use INTERFACE ADDTEMPPREFIX and INTERFACE DELTEMPPREFIX.

Monitoring network interfaces (INTERFACE statements)

To delete interfaces, stop the interfaces first. When the interfaces are stopped, they become inactive. If the TCP/IP stack is currently monitoring interfaces and detects that all monitored interfaces are inactive as a result of stopping devices, the

TCP/IP stack might issue messages regarding the problem and might trigger a recovery action. You can disable monitoring these interfaces. Specify the NOMONSYSPLEX keyword on the INTERFACE statement using the VARY TCPIP,,OBEYFILE command before stopping the interfaces. For more information about sysplex autonomics, see sysplex problem detection and recovery in z/OS Communications Server: IP Configuration Guide.

INTERFACE - IPAQENET OSA-Express QDIO interfaces statement

Use the INTERFACE statement to specify an OSA-Express QDIO Ethernet interface for IPv4.

Restriction: This statement applies to IPv4 IP addresses only.

To determine the OSA-Express microcode level, use the DISPLAY TRL command. If a specific OSA-Express function is documented with a minimum microcode level, you can use this command to determine whether that function is supported. IBM service might request the microcode level for problem diagnosis. For more information about the DISPLAY TRL command, see z/OS Communications Server: SNA Operation.

The following OSA-Express features can be defined in QDIO mode for IPv4:
- Fast Ethernet
- Gigabit Ethernet
- 1000BASE-T Ethernet
- 10G Ethernet

When you start an IPAQENET interface (and you did not specify VMAC with ROUTEALL), TCP/IP registers all non-loopback local (home) IPv4 addresses for this TCP/IP instance to the OSA-Express feature. If you subsequently add, delete, or change any home IPv4 addresses on this TCP/IP instance, TCP/IP dynamically registers the changes to the OSA-Express feature. The OSA adapter routes datagrams destined for those IPv4 addresses to this TCP/IP instance.

If a datagram is received at the OSA adapter for an unregistered IPv4 address, then the OSA-Express feature routes the datagram to the TCP/IP instance, depending on the setting of a virtual MAC (VMAC) address or definition of an instance as PRIROUTER or SECROUTER. If the datagram is not destined for a virtual MAC address and no active TCP/IP instance using this interface is defined as PRIROUTER or SECROUTER, then the OSA-Express feature discards the datagram. See the router information in z/OS Communications Server: IP Configuration Guide for more details and primary and secondary routing in z/OS Communications Server: SNA Network Implementation Guide.

For detailed instructions on setting up an OSA-Express feature, see zEnterprise System and System z10 OSA-Express Customer's Guide and Reference.

For more information about missing interrupt handler (MIH) considerations with TCP/IP interfaces, see “Missing interrupt handler factors” on page 49.

Syntax

Rule: Specify the required parameters and the CHPIDTYPE parameter in the order shown here. The OSD Interface Definition and OSX Interface Definition parameters can be specified in any order.

```
►►──INTERFace──intf_name──────────────────────────────────────────────────────────────►

                     ┌CHPIDTYPE OSD │ OSD interface definition │┐
►──┬DEFINE──IPAQENET─┤                                          ├┤ Common parameters ├┬──────►◄
   │                 └CHPIDTYPE OSX │ OSX interface definition │┘                     │
   └DELEte─────────────────────────────────────────────────────────────────────────────┘
```

OSD Interface Definition:

```
                                                                 ┌NONRouter─┐
├──PORTNAME portname──┬IPADDR──┬ipv4_address/0──────────────┬────┼──────────┼┬──────────►
                      │        ├ipv4_address────────────────┤    ├PRIRouter─┤│
                      │        └ipv4_address/num_mask_bits──┘    └SECRouter─┘│
                      └TEMPIP────────────────────────────────────────────────┘

                     ┌INBPERF BALANCED──────────────────────┐
►─┬─────────────┬────┼──────────────────────────────────────┼──────────────────────────►
  └VLANID id────┘    └INBPERF───┬DYNAMIC──┬NOWORKLOADQ──┬───┘
                                │         └WORKLOADQ────┘ │
                                ├MINCPU───────────────────┤
                                └MINLATENCY───────────────┘

                                                ┌SMCR───┐
►─┬─────────────────────────────────────────┬───┼───────┼────────────────────────────────┤
  └VMAC──┬───────────┬──┬ROUTEALL──┬────────┘   └NOSMCR─┘
         └macaddr────┘  └ROUTELCL──┘
```

OSX Interface Definition:

```
├──┬CHPID──chpid──────────┬──IPADDR ipv4_address/num_mask_bits──VLANID id──────────────►
   └PORTNAME──portname────┘

  ┌INBPERF DYNAMIC NOWORKLOADQ────────────┐   ┌VMAC ROUTEALL──────────┐
►─┼───────────────────────────────────────┼───┼───────────────────────┼────────────────┤
  └INBPERF───┬BALANCED────────────────────┤   └VMAC──┬ROUTEALL──┬─────┘
             ├MINCPU──────────────────────┤          └ROUTELCL──┘
             ├MINLATENCY──────────────────┤
             └DYNAMIC──┬NOWORKLOADQ──┬────┘
                       └WORKLOADQ────┘
```

Common parameters for OSD and OSX interface definitions:

```
                                                     ┌READSTORAGE GLOBAL──────────┐
├──┬──────────────────────────────┬──┬─────────┬─────┼────────────────────────────┼──────►
   └SOURCEVIPAINTerface ─vipa_name┘  └MTU num──┘     └READSTORAGE──┬MAX──┬────────┘
                                                                   ├AVG──┤
                                                                   └MIN──┘
```

```
                  ┌─SECCLASS 255───────────────┐ ┌─NOMONSYSPLEX─┐ ┌─NODYNVLANREG─┐
►──┬─────────┬────┼────────────────────────────┼─┼──────────────┼─┼──────────────┼──►
   └─IPBCAST─┘    └─SECCLASS security_class────┘ └─MONSYSPLEX───┘ └─DYNVLANREG───┘

   ┌─NOOLM─┐ ┌─NOISOLATE─┐
►──┼───────┼─┼───────────┼────────────────────────────────────────────────────────┤
   └─OLM───┘ └─ISOLATE───┘
```

Parameters

intf_name
: The name of the interface. The maximum length is 16 characters.

  Requirement: This name must be different than the name specified for the PORTNAME parameter.

DEFINE
: Specifies that this definition is to be added to the list of defined interfaces.

DELETE
: Specifies that this definition is to be deleted from the list of defined interfaces. The *intf_name* value must be the name of an interface previously defined by an INTERFACE statement. Specifying INTERFACE DELETE deletes the home IP address for the interface.

IPAQENET
: Indicates that the interface uses the interface based on IP assist, which belongs to the QDIO family of interfaces, and uses the Ethernet protocol.

CHPIDTYPE
: An optional parameter indicating the CHPID type of the OSA-Express QDIO interface.

  OSD Indicates an external data network type. This is the default value.

  OSX The intraensemble data network. See z/OS Communications Server: IP Configuration Guide for information about requirements necessary to make an OSX work.

  Rule: You must specify an OSD interface definition to make this interface eligible to use Shared Memory Communications over Remote Direct Memory Access (SMC-R).

CHPID *chpid*
: This parameter applies only to interfaces of CHPIDTYPE OSX and is used to specify the CHPID for the interface. This value is a 2-character hexadecimal value (00 - FF).

PORTNAME *portname*
: Use this parameter to specify the PORT name that is in the TRLE definition for the QDIO interface. The TRLE must be defined as MPCLEVEL=QDIO. For details about defining a TRLE, see z/OS Communications Server: SNA Resource Definition Reference.

  Requirement: The *portname* value must be different than the name specified for *intf_name*.

IPADDR

  ipv4_address
  : The home IP address for this interface.

Requirement: The IP address must be specified in dotted decimal form.

num_mask_bits

An integer value in the range 0 - 32 that represents the number of leftmost significant bits for the subnet mask of the interface. This value also controls how ARP processing for VIPAs is handled for this interface. When you specify a nonzero value, the TCP/IP stack informs OSA to perform ARP processing for a VIPA only if the VIPA is configured in the same subnet as the OSA (as defined by this subnet mask). The default is 0 for CHPIDTYPE OSD. This parameter is required for CHPIDTYPE OSX..

Requirement: If you are configuring multiple IPv4 VLAN interfaces to the same OSA-Express feature, then you must specify a nonzero value for the *num_mask_bits* variable for each of these interfaces and the resulting subnet must be unique for each of these interfaces.

Rule: If you are using OMPROUTE and OMPROUTE is not configured to ignore this interface, ensure that the subnet mask value that you define on this parameter matches the subnet mask used by OMPROUTE for this interface. The subnet mask used by OMPROUTE is the subnet mask value defined on the corresponding OMPROUTE statement (OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE) for this interface. If no OMPROUTE statement is specified for this interface, the subnet mask used by OMPROUTE is the class mask for the interface IP address.

TEMPIP

Specifies that the interface starts with an IP address of 0.0.0.0. The interface can be used for broadcast traffic. This parameter applies only to interfaces that are defined with CHPIDTYPE OSD.

Guideline: Use TEMPIP interfaces in a unit test environment to support an application that provides a DHCP client, such as IBM Rational® Developer for System z Unit Test feature (Rdz-UT). For more information about configuring a TEMPIP interface, see Using TEMPIP interfaces in z/OS Communications Server: IP Configuration Guide.

NONROUTER

If a datagram is received at this interface for an unknown IP address, the datagram is not routed to this TCP/IP instance. This is the default value.

The PRIROUTER and SECROUTER parameters interact with the VLANID parameter. See the VLANID parameter definition to understand this relationship.

For more information about VLANID parameter interactions, see z/OS Communications Server: IP Configuration Guide.

Rule: This keyword applies only to interfaces of CHPIDTYPE OSD and is ignored if the VMAC parameter is configured on the INTERFACE statement.

PRIROUTER

If a datagram is received at this interface for an unknown IP address and is not destined for a virtual MAC, the datagram is routed to this TCP/IP instance. This parameter interacts with the VLANID parameter. See the VLANID parameter definition to understand this relationship.

For more information about VLANID parameter interactions, see z/OS Communications Server: IP Configuration Guide.

Rule: This keyword applies only to interfaces of CHPIDTYPE OSD and is ignored if the VMAC parameter is configured on the INTERFACE statement.

SECROUTER

If a datagram is received at this interface for an unknown IP address and is not destined for a virtual MAC, and there is no active TCP/IP instance defined as PRIROUTER, then the datagram is routed to this TCP/IP instance. This parameter interacts with the VLANID parameter. See the VLANID parameter definition to understand this relationship.

For more information about VLANID parameter interactions, see z/OS Communications Server: IP Configuration Guide.

Rule: This keyword applies only to interfaces of CHPIDTYPE OSD and is ignored if the VMAC parameter is configured on the INTERFACE statement.

VLANID id

Specifies the decimal virtual LAN identifier to be assigned to the OSA-Express interface. This field should be a virtual LAN identifier recognized by the switch for the LAN that is connected to this OSA-Express interface. The valid range is 1 - 4094. This parameter is optional for CHPIDTYPE OSD and required for CHPIDTYPE OSX.

Guideline: Installation configuration on other platforms or related to Ensemble networking can limit the maximum VLANID of 4096.

The VLANID parameter interacts with the PRIROUTER and SECROUTER parameters. If you configure both the VLANID parameter and either PRIROUTER or SECROUTER parameter, then this TCP/IP instance acts as a router for this VLAN (ID) only. Datagrams that are received at this device instance for an unknown IP address and are not destined for a virtual MAC are routed only to this TCP/IP instance if it is VLAN tagged with this VLAN ID. For more information about VLANID parameter interactions, see z/OS Communications Server: IP Configuration Guide.

Rule: If you are configuring multiple VLAN interfaces to the same OSA-Express feature, then you must specify the VMAC parameter (with the default ROUTEALL attribute) on the INTERFACE statement for each of these interfaces.

Restriction: The stack supports a maximum of 32 IPv4 VLAN interfaces to the same OSA-Express port. Additional VLANID limitations might exist if this interface can be used with Shared Memory Communications over Remote Direct Memory Access (SMC-R). See VLANID considerations in z/OS Communications Server: IP Configuration Guide for details.

INBPERF

An optional parameter that indicates how processing of inbound traffic for the QDIO interface is performed.

There are three supported static settings that indicate how frequently the adapter should interrupt the host for inbound traffic: BALANCED, MINCPU, and MINLATENCY. The static settings use static interrupt-timing values. The static values are not always optimal for all workload types or traffic patterns, and the static values cannot account for changes in traffic patterns.

There is also one supported dynamic setting (DYNAMIC). This setting causes the host (stack) to dynamically adjust the timer-interrupt value while the device is active and in use. This function exploits an OSA hardware function called Dynamic LAN Idle. Unlike the static settings, the DYNAMIC setting reacts to changes in traffic patterns and sets the interrupt-timing values to maximize throughput. The dynamic setting does not incur additional CPU consumption that might be produced when you specify any of the static settings. In addition, the DYNAMIC setting uses the OSA Dynamic Router Architecture function to enable QDIO inbound workload queues for specific inbound traffic types.

Result: When you specify OLM on the INTERFACE statement, the INBPERF parameter is ignored and the statement takes the value DYNAMIC.

Valid values for INBPERF are:

BALANCED
This setting uses a static interrupt-timing value, which is selected to achieve reasonably high throughput and reasonably low CPU consumption. This is the default value for CHPIDTYPE OSD. .

DYNAMIC
This setting causes the host to dynamically signal the OSA-Express feature to change the timer-interrupt value, based on current inbound workload conditions. The DYNAMIC setting is effective only for OSA-Express2 or later features on at least an IBM System z9 that supports the corresponding Dynamic LAN Idle function. See the 2097DEVICE Preventive Service Planning (PSP) bucket for more information about the OSA-Express3 adapter that supports this function. The DYNAMIC setting should outperform the other three static settings for most workload combinations. This is the default value for CHPIDTYPE OSX.

If the DYNAMIC setting is specified for an OSA-Express adapter that does not support the dynamic LAN Idle function, the stack reverts to using the BALANCED setting.

WORKLOADQ | NOWORKLOADQ

This subparameter controls the QDIO inbound workload queueing function for the interface. QDIO inbound workload queueing is effective only for OSA-Express features in QDIO mode that support the corresponding Data Router Architecture. OSA-Express features that support workload queueing do not necessarily support workload queueing for all possible traffic types. For more information about the QDIO inbound workload queueing function and the OSA-Express features that support it, see QDIO inbound workload queueing in z/OS Communications Server: IP Configuration Guide.

NOWORKLOADQ
Specifies that QDIO inbound workload queueing is not enabled for inbound traffic. All inbound traffic for this interface uses a single input queue. This is the default value.

WORKLOADQ
Specifies that QDIO inbound workload queueing is enabled for inbound traffic.

If the WORKLOADQ subparameter is specified, QDIO inbound workload queueing is enabled for specific inbound traffic types. A primary input queue is reserved for all other traffic types.

Ancillary input queues (AIQs) are created for the following inbound traffic types when supported by the OSA-Express feature:

- Sysplex distributor
- Streaming workloads (for example FTP)
- Enterprise Extender (EE)

Requirement: You must specify the VMAC parameter with WORKLOADQ to enable QDIO inbound workload queueing.

If the WORKLOADQ setting is specified for an OSA-Express adapter that does not support the Data Router Architecture function, the stack reverts to using a single input queue.

MINCPU
This setting uses a static interrupt-timing value, which is selected to minimize host interrupts without regard to throughput. This mode of operation might result in minor queueing delays (latency) for packets flowing into the host, which is not optimal for workloads with demanding latency requirements.

MINLATENCY
This setting uses a static interrupt-timing value, which is selected to minimize latency (delay), by more aggressively sending received packets to the host. This mode of operation generally results in higher CPU consumption than the other three settings. Use this setting only if host CPU consumption is not an issue.

`VMAC` *macaddr*
Specifies the virtual MAC address, which can be represented by 12 hexadecimal characters. The OSA-Express device uses this address rather than the physical MAC address of the device for all IPv4 packets sent to and received from this TCP/IP stack. For CHPIDTYPE OSD, using a virtual MAC address is optional. For CHPIDTYPE OSX, using a virtual MAC address is required, so the VMAC parameter is the default

The *macaddr* value is optional. The *macaddr* value is optional for CHPIDTYPE OSD and cannot be specified for CHPIDTYPE OSX. If you do not code the *macaddr* value, then the OSA-Express device generates a virtual MAC address. If you do code the *macaddr* value, it must be defined as a locally administered individual MAC address. This means the MAC address must have bit 6 (the universal or local flag U bit) of the first byte set to 1 and bit 7 (the group or individual flag G bit) of the first byte set to 0. The second hexadecimal character must be 2, 6, A, or E. The bit positions within the 12 hexadecimal characters are indicated as follows:

```
|               1|1              3|3              4|
|0              5|6              1|2              7|
+----------------+----------------+----------------+
|xxxxxxUGxxxxxxxx|xxxxxxxxxxxxxxxx|xxxxxxxxxxxxxxxx|
+----------------+----------------+----------------+
```

Rules:

- The same virtual MAC address generated by the OSA-Express device during interface activation remains in effect for this OSA-Express for this TCP/IP stack, even if the interface is stopped or becomes inoperative (INOPs). A new virtual MAC address is generated only if the INTERFACE statement is deleted and redefined or if the TCP/IP stack is recycled.

- The NONROUTER, PRIROUTER, and SECROUTER parameters are ignored for an OSA-Express interface if the VMAC parameter is configured on the INTERFACE statement.

Guideline: Unless the virtual MAC address representing this OSA-Express device must remain the same even after TCP/IP termination and restart, configure VMAC without a *macaddr* value and allow the OSA-Express device to generate it. This guarantees that the VMAC address is unique from all other physical MAC addresses and from all other VMAC addresses generated by any OSA-Express feature.

ROUTEALL
Specifies that all IP traffic destined to the virtual MAC is forwarded by the OSA-Express device to the TCP/IP stack. This is the default value. See the router information in z/OS Communications Server: IP Configuration Guide for more details.

ROUTELCL
Specifies that only traffic destined to the virtual MAC and whose destination IP address is registered with the OSA-Express device by this TCP/IP stack is forwarded by the OSA-Express. See the router information in z/OS Communications Server: IP Configuration Guide for more details.

SMCR | NOSMCR
Specifies whether this interface can be used with Shared Memory Communications over Remote Direct Memory Access (SMC-R) for external data network communications.

NOSMCR
Specifies that this interface cannot be used for new TCP connections with SMC-R for external data network communications.

SMCR
Specifies that this interface can be used for new TCP connections with SMC-R for external data network communications. This is the default setting.

Rules:
- SMCR and NOSCMR are valid with CHPIDTYPE OSD definitions only.
- SMCR has no effect unless at least one Peripheral Component Interconnect Express (PCIe) function ID (PFID) value is specified by using the PFID subparameter of the SMCR parameter on the GLOBALCONFIG statement.
- SMCR has no effect unless a nonzero subnet mask is configured on the INTERFACE statement.

SOURCEVIPAINTERFACE *vipa_name*
Specifies which previously-defined static VIPA interface is used for SOURCEVIPA (when IPCONFIG SOURCEVIPA is in effect). The *vipa_name* value is the interface name (or link name) for a VIRTUAL interface. This parameter is optional.

Requirement: The VIRTUAL interface must be defined prior to specifying this INTERFACE statement to the TCP/IP stack. It must either already be defined, or the INTERFACE statement (or DEVICE and LINK statements) that define the static VIPA must precede this INTERFACE statement in the profile data set.

Tip: The SOURCEVIPAINTERFACE setting can be overridden. See the information about Source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

MTU *num*

The maximum transmission unit (MTU), in bytes. This value can be in the range 576 - 8992. The minimum MTU for IPv4 is 576. The stack takes the minimum of the configured value and the value supported by the device (returned by OSA).

The MTU default, which depends on the value that is supported by the device, is the following value:

- Gigabit Ethernet default MTU = 8992
- Fast Ethernet default MTU = 1492

The MTU default is 1492 for Fast Ethernet; otherwise, it is 8992.

Rule: If you are using OMPROUTE and OMPROUTE is not configured to ignore this interface, ensure that the MTU that you define on this parameter matches the MTU used by OMPROUTE for this interface. The MTU used by OMPROUTE is the MTU value defined on the corresponding OMPROUTE statement (OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE) for this interface. If an MTU value is not defined on the corresponding OMPROUTE statement for this interface or if no OMPROUTE statement is specified for this interface, the MTU used by OMPROUTE is the minimum MTU for IPv4 (576).

Tip: See z/OS Communications Server: IP Configuration Guide, in section Maximum transmission unit considerations, for additional information about how TCP/IP uses the MTU to determine the largest size frame to send.

READSTORAGE

An optional parameter indicating the amount of fixed storage that z/OS Communications Server should keep available for read processing for this adapter. Use the QDIOSTG VTAM start option to specify a value that applies to all OSA-Express adapters in QDIO mode. You can use the READSTORAGE keyword to override the global QDIOSTG value for this adapter based on the inbound workload that you expect over this interface on this stack. The valid values for READSTORAGE are:

GLOBAL
The amount of storage is determined by the QDIOSTG VTAM start option. This is the default value.

MAX Use this value if you expect a heavy inbound workload over this interface.

AVG Use this value if you expect a medium inbound workload over this interface.

MIN Use this value if you expect a light inbound workload over this interface.

Tip: See the description of the QDIOSTG VTAM start option in the z/OS Communications Server: SNA Resource Definition Reference for details about exactly how much storage is allocated by z/OS Communications Server for each of these values.

IPBCAST

Specifies that the interface both sends and receives IP broadcast packets. If this parameter is not specified, no IP broadcast packets are sent or received on this interface.

SECCLASS *security_class*

Use this parameter to associate a security class for IP filtering with this interface. For traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. You can specify filter rules in the TCP/IP profile or in an IP security policy file that is read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSECRULE statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX

Specifies whether sysplex autonomics should monitor the interface's status.

NOMONSYSPLEX

Specifies that sysplex autonomics should not monitor the interface's status. This is the default value.

MONSYSPLEX

Specifies that sysplex autonomics should monitor the interface's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this interface is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

DYNVLANREG | NODYNVLANREG

This parameter controls whether or not the VLAN ID for this interface is dynamically or statically registered with the physical switch on the LAN.

Restriction: This parameter is applicable only if a VLAN ID is specified on the statement.

Dynamic registration of VLAN IDs is handled by the OSA-Express feature and the physical switch on your LAN. Therefore, in order for the DYNVLANREG parameter to be effective, both must be at a level that provides the necessary hardware support for dynamic VLAN ID registration. After the interface is active, you can view the Netstat DEvlinks/**-d** report output to determine whether your OSA-Express feature can support VLAN dynamic registration. This Netstat report also displays whether dynamic VLAN ID registration has been configured for the interface.

NODYNVLANREG

Specifies that if a VLAN ID is configured for this interface, it must be manually registered with the physical switches on the corresponding LAN. This is the default value. If this parameter is specified without a VLAN ID, then it is ignored.

DYNVLANREG

Specifies that if a VLAN ID is configured for this interface, it is dynamically registered with the physical switches on the corresponding LAN. If this parameter is specified without a VLAN ID, then warning message EZZ0056I is issued and the NODYNVLANREG setting is used instead.

OLM | NOOLM

An optional parameter indicating whether an OSA-Express adapter operates in optimized latency mode.

NOOLM

Specifies that the OSA-Express adapter should not operate in optimized latency mode. This is the default value.

OLM Specifies that the OSA-Express adapter operates in optimized latency mode (OLM). Optimized latency mode optimizes interrupt processing for both inbound and outbound data. Use this mode for workloads that have demanding latency requirements. Because this mode can provide significant increases of throughput, particularly for interactive, non-streaming workloads. For more information about optimized latency mode, see the optimized latency mode topic in z/OS Communications Server: IP Configuration Guide.

Guidelines:

- Because of the operating characteristics of optimized latency mode, you might need to change your configuration to direct traffic to particular OSA-Express write priority queues and to limit the number of concurrent users sharing an OSA-Express configured for optimized latency mode. For more information about OLM, see the optimized latency mode topic in z/OS Communications Server: IP Configuration Guide.
- The optimized latency mode function targets a z/OS environment with a high-volume, interactive workloads. Although optimized latency mode can compensate for some mixing of workloads, an excessive amount of high-volume streaming workloads, such as bulk data or file transfer, can result in higher CPU consumption.

Restrictions:

- This function is limited to OSA-Express3 or later Ethernet features in QDIO mode that are running with an IBM System z10 or later. See the 2097 DEVICE Preventive Service Planning (PSP) bucket for more information.
- Traffic that is either inbound over or being forwarded to an OSA-Express configured to use optimized latency mode is not eligible for the accelerated routing function provided by HiperSockets Accelerator and QDIO Accelerator.
- For an OSA-Express configured to use optimized latency mode, the stack ignores the configured or default INBPERF setting and uses the value DYNAMIC.

ISOLATE | NOISOLATE

Specifies whether packets should be directly routed between TCP/IP stacks that share the OSA adapter.

NOISOLATE

Route packets directly between TCP/IP stacks sharing the OSA adapter. In this mode, if the next hop address was registered by another stack that is sharing the OSA adapter, then the OSA-Express adapter routes the packet directly to the sharing stack without putting the packet on the external LAN.

ISOLATE

Prevent OSA-Express from routing packets directly to another TCP/IP stack that is sharing the OSA adapter. In this mode, OSA-Express adapter discards any packets when the next hop address was registered by another stack that is sharing the OSA adapter. Packets can flow between two stacks that share the OSA only by first going through a router on the LAN. For more details, see the OSA-Express connection isolation information in z/OS Communications Server: IP Configuration Guide.

Tips:

- If you isolate an interface, there might be an adverse effect on latency.
- You can selectively apply OSA-Express connection isolation to individual virtual LANs.
- The OSA-Express adapter requires that both stacks sharing the port be non-isolated for direct routing to occur. Therefore, for traffic between two stacks sharing the OSA adapter, as long as at least one of the stacks is isolated, connection isolation is in effect for traffic in both directions between these stacks.

Restriction: This function is limited to OSA-Express2 or later Ethernet features in QDIO mode and running at least an IBM System z9 Enterprise Class (EC) or z9 Business Class (BC). See the 2094DEVICE, 2096DEVICE, 2097DEVICE, or 2098DEVICE Preventive Service Planning (PSP) bucket for more information.

Steps for modifying

See "Summary of INTERFACE statements" on page 141 for modification information.

Examples

```
INTERFACE OSAQDIO24
DEFINE IPAQENET
PORTNAME OSAQDIO2
SOURCEVIPAINT VIPAV4
IPADDR 100.1.1.1/24
```

Related topics

- "BEGINROUTES statement" on page 28
- "BSDROUTINGPARMS statement" on page 36
- "DEVICE and LINK — MPCIPA OSA-Express QDIO devices statement" on page 74
- "GLOBALCONFIG statement" on page 117
- "INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement" on page 161
- "SACONFIG statement" on page 271
- "START statement" on page 292
- "STOP statement" on page 293

INTERFACE — IPAQIDIO HiperSockets interfaces statement

Use the INTERFACE statement for IPAQIDIO to configure IPv4 HiperSockets connectivity. Use the CHPID parameter to specify the value of the desired IQD CHPID that was configured within HCD. HiperSockets interfaces do not require a corresponding TRLE definition. Instead, the TRLE is dynamically built when the interface is started.

To determine the HiperSockets microcode level, use the DISPLAY TRL command. If a specific HiperSockets function is documented with a minimum microcode level, you can use this command to determine whether that function is supported. IBM service might request the microcode level for problem diagnosis. For more information, see DISPLAY TRL command in z/OS Communications Server: SNA Operation.

Rule: Specify the required parameters in the order shown here. The Interface Definition parameters can be specified in any order.

Syntax

```
►►──INTERFace──intf_name──┬─DEFINE──IPAQIDIO──┤ Interface Definition ├─┬──────────►◄
                          └─DELEte─────────────────────────────────────┘
```

Interface Definition:

```
├──CHPID chpid──IPADDR──┬─ipv4_address/0─────────────┬──┬─────────┬──────────────►
                        ├─ipv4_address───────────────┤  └─IPBCAST─┘
                        └─ipv4_address/num_mask_bits─┘

     ┌─READSTORAGE GLOBAL─────┐
►──┬─────────┬──┼────────────────────────┼──┬───────────┬──────────────────────────►
   └─MTU num─┘  └─READSTORAGE──┬─MAX─┬───┘  └─VLANID id─┘
                               ├─AVG─┤
                               └─MIN─┘

                                        ┌─SECCLASS 255────────────┐
►──┬──────────────────────────────┬──┼─────────────────────────┼───────────────────►
   └─SOURCEVIPAINTerface vipa_name─┘  └─SECCLASS security_class─┘

   ┌─NOMONSYSPLEX─┐
►──┼──────────────┼────────────────────────────────────────────────────────────────┤
   └─MONSYSPLEX───┘
```

Parameters

intf_name
    The name of the interface. The maximum length is 16 characters.

DEFINE
    Specifies that this definition is to be added to the list of defined interfaces.

DELETE
    Specifies that this definition is to be deleted from the list of defined interfaces.

The *intf_name* must be the name of an interface that was previously defined by an INTERFACE statement. INTERFACE DELETE deletes the home IP address for the interface.

IPAQIDIO
Indicates that the interface is for HiperSockets IPv4.

CHPID *chpid*
Use this parameter to specify the IQD CHPID for the HiperSockets interface. This value is a 2-character hexadecimal value (00x - FFx). The hexadecimal value specified on the CHPID parameter cannot be the same value that is used for the dynamic XCF HiperSockets interface. See the IQDCHPID start option in the z/OS Communications Server: SNA Resource Definition Reference.

IPADDR *ipaddr_spec*

ipv4_address
The home IP address for this interface.

Requirement: The IP address must be specified in dotted decimal form.

num_mask_bits
An integer value in the range 0 - 32 that represents the number of leftmost significant bits for the subnet mask of the interface. The default is 0.

Requirement: If you are configuring multiple IPv4 VLAN interfaces to the same HiperSockets CHPID, then you must specify a nonzero value for the *num_mask_bits* variable for each of these interfaces and the resulting subnet must be unique for each of these interfaces.

Rule: If you are using OMPROUTE and OMPROUTE is not configured to ignore this interface, ensure that the subnet mask value that you define on this parameter matches the subnet mask used by OMPROUTE for this interface. The subnet mask used by OMPROUTE is the subnet mask value defined on the corresponding OMPROUTE statement (OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE) for this interface. If no OMPROUTE statement is specified for this interface, the subnet mask used by OMPROUTE is the class mask for the interface IP address.

IPBCAST
Specifies that the interface both sends and receives IP broadcast packets. If this parameter is not specified, no IP broadcast packets are sent or received on this interface.

MTU *num*
The maximum transmission unit (MTU), in bytes. This value can be in the range 576 - 57344. The minimum MTU for IPv4 is 576. The stack takes the minimum of the configured value and the firmware supported value (which is the IQD frame size configured in HCD minus 8192)

The MTU default is equal to the IQD frame size minus 8192.

Rule: If you are using OMPROUTE and OMPROUTE is not configured to ignore this interface, ensure that the MTU that you define on this parameter matches the MTU used by OMPROUTE for this interface. The MTU used by OMPROUTE is the MTU value defined on the corresponding OMPROUTE statement (OSPF_INTERFACE, RIP_INTERFACE, or INTERFACE) for this interface. If an MTU value is not defined on the corresponding OMPROUTE statement for this interface or if no OMPROUTE statement is specified for this interface, the MTU used by OMPROUTE is the minimum MTU for IPv4 (576).

Tip: See Determining the maximum transmission unit in z/OS Communications Server: IP Configuration Guide for more information about how TCP/IP uses the MTU to determine the largest size frame to send.

READSTORAGE
An optional parameter that indicates the amount of fixed storage that z/OS CS should keep available for read processing for this interface. The IQDIOSTG VTAM start option allows you to specify a value that applies to all HiperSockets interfaces. You can use the READSTORAGE keyword to override the global IQDIOSTG value for this interface based on the inbound workload that you expect over this interface on this stack. The following values are valid:

GLOBAL
The amount of storage is determined by the IQDIOSTG VTAM start option. This is the default value.

MAX
Use this value if you expect a heavy inbound workload over this interface.

AVG
Use this value if you expect a medium inbound workload over this interface.

MIN
Use this value if you expect a light inbound workload over this interface.

Tip: See the description of the IQDIOSTG VTAM start option in the z/OS Communications Server: SNA Resource Definition Reference for details about exactly how much storage is allocated by z/OS Communications Server for each of these values.

VLANID *id*
An optional parameter followed by a decimal number indicating the virtual LAN identifier to be assigned to this HiperSockets interface. The valid range is 1 - 4 094.

SOURCEVIPAINTERFACE *vipa_name*
An optional parameter used to specify which previously-defined VIPA interface is to be used for SOURCEVIPA (when IPCONFIG SOURCEVIPA is in effect). The *vipa_name* value is the interface name for a VIRTUAL interface.

Requirement: The VIRTUAL interface or the link must be defined prior to specifying this INTERFACE statement to the TCP/IP stack. It must either already be defined, or the INTERFACE statement (or DEVICE and LINK statements) that define the static VIPA must precede this INTERFACE statement in the profile data set.

Tip: The use of the SOURCEVIPAINTERFACE parameter can be overridden. See the information about Source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. For traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file that is read by the Policy Agent. Filter rules can include a security class specification

on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSECRULE statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. See security class values in z/OS Communications Server: IP Configuration Guide for more information.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the interface's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the interface's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor interface's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over the interface is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

Steps for modifying

See “Summary of INTERFACE statements” on page 141 for modification information.

Examples

```
INTERFACE HIPERSOCK1 DEFINE IPAQIDIO CHPID FC
             IPADDR 9.1.1.1/24
```

Related topics

- “BEGINROUTES statement” on page 28
- “DEVICE and LINK — MPCIPA HiperSockets devices statement” on page 85
- “INTERFACE — IPAQIDIO6 HiperSockets interfaces statement” on page 177
- “START statement” on page 292
- “STOP statement” on page 293

INTERFACE — VIRTUAL interfaces statement

Use the INTERFACE statement to specify a static virtual interface.

You can define multiple virtual IPv4 addresses on one TCP/IP image by specifying multiple VIRTUAL INTERFACE statements.

Syntax

```
►►──INTERFace intf_name──┬─DEFINE──VIRTUAL──┤ Interface Definition ├─┬──────────►◄
                         └─DELEte────────────────────────────────────┘
```

Interface Definition:

```
├──IPADDR── ipv4_address──────────────────────────────────────────────────┤
```

Parameters

intf_name
The name of the interface. The maximum length is 16 characters.

DEFINE
Specifies that this definition is to be added to the list of defined interfaces.

DELETE
Specifies that this definition is to be deleted from the list of defined interfaces. The *intf_name* must be the name of an interface previously defined by an INTERFACE statement. INTERFACE DELETE deletes all home IP addresses for the interface.

VIRTUAL
Indicates that the interface is not associated with real hardware and is used for fault tolerance support.

IPADDR *ipv4_address*
This parameter is required and must be one IPv4 address specified in dotted decimal form.

Steps for modifying

See "Summary of INTERFACE statements" on page 141 for modification information.

Examples

```
INTERFACE VIPAV4 DEFINE
  VIRTUAL
  IPADDR 9.1.1.1
```

Usage notes

- The TCP/IP stack does not maintain interface counters for VIRTUAL interfaces.
- A VIRTUAL interface name cannot be coded in the BEGINROUTES block.

Related topics

"IPCONFIG statement" on page 190

INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement

Use the INTERFACE statement to specify an OSA-Express QDIO Ethernet interface for IPv6.

To determine the OSA-Express microcode level, use the DISPLAY TRL command. If a specific OSA-Express function is documented with a minimum microcode level, you can use this command to determine whether that function is supported. IBM service might request the microcode level for problem diagnosis. For more information about the DISPLAY TRL command, see z/OS Communications Server: SNA Operation.

The following OSA-Express features can be defined in QDIO mode for IPv6:

- Fast Ethernet
- Gigabit Ethernet
- 1000BASE-T Ethernet
- 10G Ethernet

When you start an IPAQENET6 interface (and you do not specify VMAC with ROUTEALL), TCP/IP registers all non-loopback local (home) IPv6 addresses for this TCP/IP instance to the OSA-Express feature. If you subsequently add, delete, or change any home IPv6 addresses on this TCP/IP instance, TCP/IP dynamically registers the changes to the OSA-Express feature. If stateless address autoconfiguration is enabled for this interface, TCP/IP dynamically registers autoconfigured addresses to the OSA-Express feature. This includes both public and temporary autoconfigured addresses. The OSA-Express feature routes datagrams destined to those IPv6 addresses to this TCP/IP instance.

If a datagram is received by the OSA adapter for an unregistered IPv6 address, then the OSA-Express feature routes the datagram to the TCP/IP instance, depending on the setting of a virtual MAC (VMAC) address or whether the definition of an instance is PRIROUTER or SECROUTER. If the datagram is not destined for a virtual MAC address and no active TCP/IP instance using this interface is defined as PRIROUTER or SECROUTER, then the OSA-Express feature discards the datagram. For more details about the OSA-Express feature routing considerations, see the router information in z/OS Communications Server: IP Configuration Guide and primary and secondary routing in z/OS Communications Server: SNA Network Implementation Guide.

For detailed instructions on setting up an OSA-Express feature, see zEnterprise System and System z10 OSA-Express Customer's Guide and Reference.

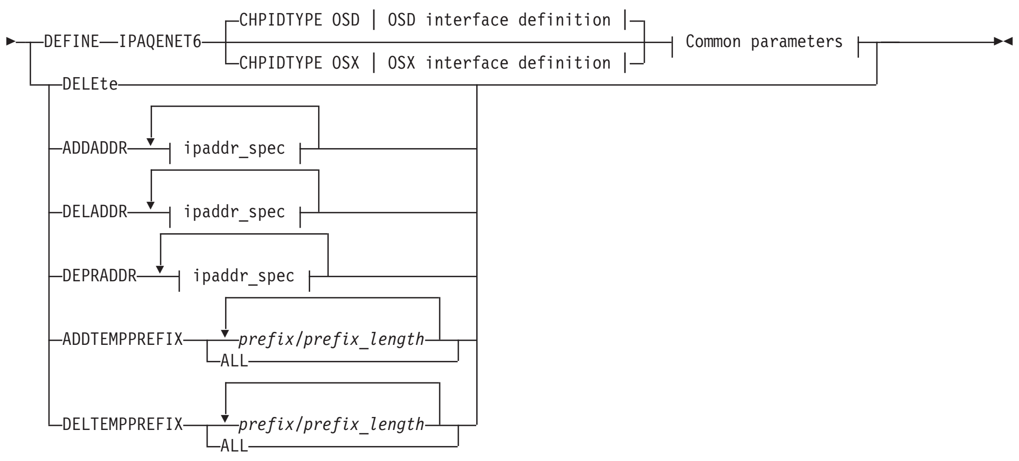
For more information about missing interrupt handler (MIH) considerations with TCP/IP interfaces, see “Missing interrupt handler factors” on page 49.

Restriction: This statement applies to IPv6 IP addresses only.

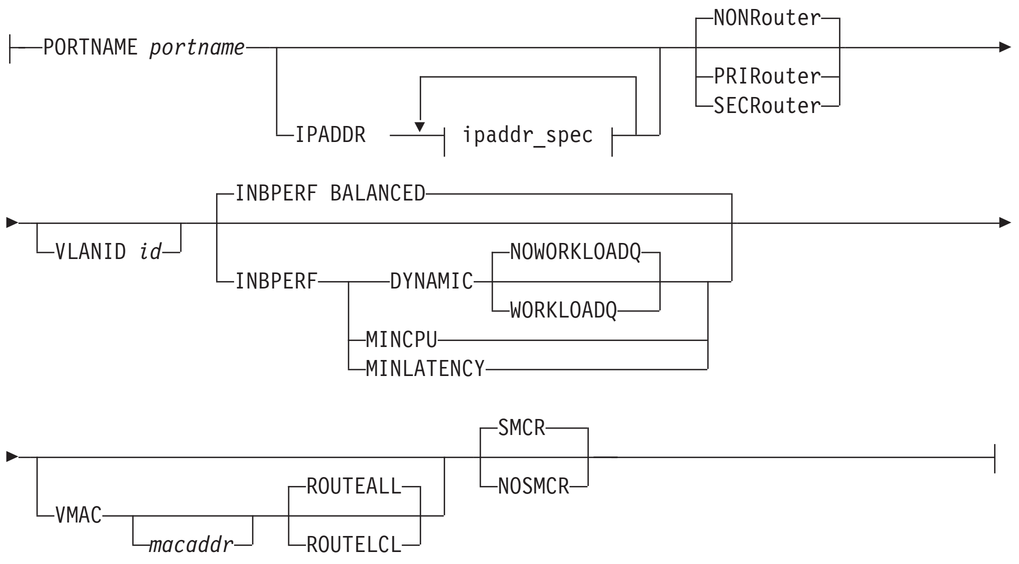
Syntax

Rule: Specify the required parameters and the CHPIDTYPE parameter in the order shown here. The OSD Interface Definition and OSX Interface Definition parameters can be specified in any order.

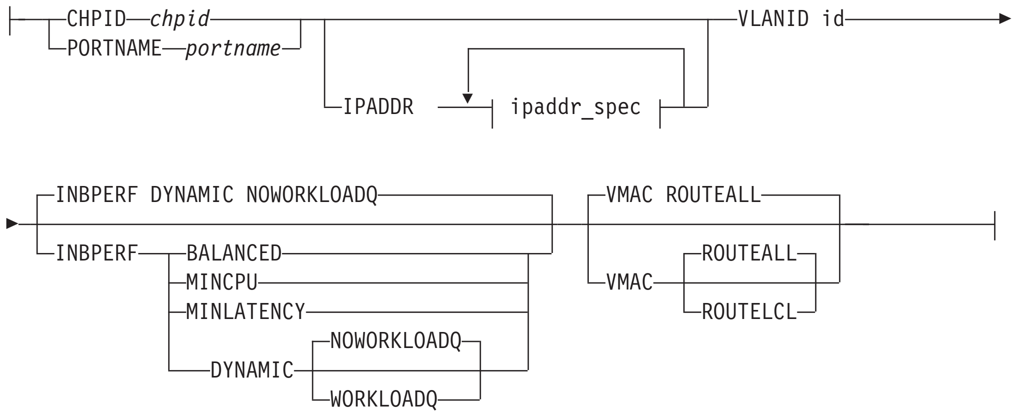
```
►►──INTERFace──intf_name──────────────────────────────────────────────────────►
```



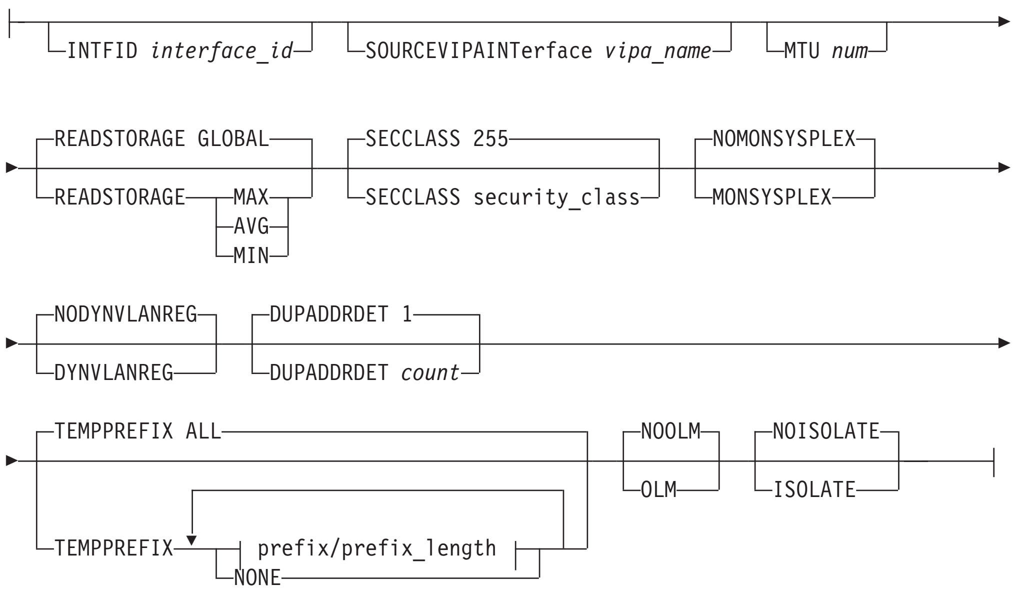
OSD interface definition:



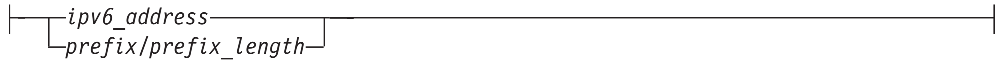
OSX Interface definition:



Common parameters for OSD and OSX interface definitions:



ipaddr_spec:



Parameters

intf_name
The name of the interface. The maximum length is 16 characters.

Requirement: This name must be different than the name specified for the PORTNAME parameter.

Restriction: Do not specify the value PUBLICADDRS or TEMPADDRS for the interface name. The values PUBLICADDRS and TEMPADDRS are keywords on the SRCIP statement. These values are not recognized if they are specified as an IPv6 interface name on an SRCIP entry.

DEFINE
Specifies that this definition is to be added to the list of defined interfaces.

DELETE
Specifies that this definition is to be deleted from the list of defined interfaces. The *intf_name* must be the name of an interface previously defined by an INTERFACE statement. INTERFACE DELETE deletes all home IP addresses for the interface.

CHPIDTYPE
An optional parameter indicating the CHPID type of the OSA-Express QDIO interface.

OSD The external data network. This is the default value.

OSX The intraensemble data network. See z/OS Communications Server: IP Configuration Guide for information about requirements necessary to make an OSX work.

Rule: You must specify an OSD interface definition to make this OSA-Express QDIO interface use Shared Memory Communications over Remote Direct Memory Access (SMC-R) for external data network communications.

IPADDR *ipaddr_spec*
For information about the IPv6 address restrictions, see "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143.

The following value can be specified for *ipaddr_spec*:

ipv6_address

This parameter can be one of the following values:
- ipv6_addr (A fully qualified IPv6 address is in colon-hexadecimal format.)
- prefix/64 [The digits (in colon-hexadecimal format) before the / represent the prefix. The prefix length represents the length of the prefix in bits. If a prefix length is coded, it must be equal to 64. When a prefix is specified, TCP/IP constructs the IPv6 address by appending the interface ID to it.]

Restriction: If you code a prefix that is longer than 64 bits, it is truncated to 64 bits, and no error messages are issued.

ADDADDR *ipaddr_spec*
Allows the addition of IP addresses to an existing INTERFACE definition (similar to updating the HOME list with the VARY TCPIP,,OBEYFILE command) without having to delete and redefine the INTERFACE. This can be used to change the autoconfiguration state of an interface. If ADDADDR is coded and this is the first manually configured IP address for the interface, then TCP/IP disables autoconfiguration for the interface. The *intf_name* coded with ADDADDR must be the name of an interface previously defined by an INTERFACE statement.

Any public or temporary addresses that had previously been autoconfigured for the interface are deleted.

DELADDR *ipaddr_spec*
Allows you to delete IP addresses from an existing INTERFACE definition. If DELADDR is coded for the last or only manually configured IP address for an interface, then TCP/IP enables autoconfiguration for the interface. DELADDR is valid only for an IP address or prefix configured manually. The *intf_name* coded with DELADDR must be the name of an interface previously defined by an INTERFACE statement. DELADDR is valid only in a data set specified on a VARY TCPIP,,OBEYFILE command.

Guideline: If you specify a prefix for DELADDR, then the only IP addresses affected are those defined by way of the same prefix specified on IPADDR or ADDADDR.

DEPRADDR *ipaddr_spec*
The DEPRADDR keyword allows you to deprecate an IP address. This can assist with site renumbering. DEPRADDR is valid only for an IP address or prefix configured manually. If you use DEPRADDR to deprecate an IP address, you can subsequently use ADDADDR again to make that IP address preferred. For DEPRADDR, the *interface_name* must be the name of an interface previously defined by an INTERFACE statement. DEPRADDR is valid only in a data set specified on a VARY TCPIP,,OBEYFILE command.

Guideline: If you specify a prefix for DEPRADDR, then the only IP addresses affected are those defined by way of the same prefix specified on IPADDR or ADDADDR.

ADDTEMPPREFIX

Use the ADDTEMPPREFIX keyword to add prefixes to the temporary prefixes list of an existing INTERFACE definition without having to delete and redefine the INTERFACE statement. The temporary prefixes list limits the set of prefixes for which temporary IPv6 addresses can be generated. A temporary IPv6 address is generated when a router advertisement containing the prefix is processed, and the prefix is included in one of the prefixes in the temporary prefixes list. For example, if the temporary prefixes list for an interface contains a single prefix 2001:0db8:58cd::/48, a temporary address is generated for advertised prefix 2001:0db8:58cd:0001/64; however, a temporary address is not generated on this interface for advertised prefix 2001:0db8:5555:0001/64. The *intf_name* variable coded with ADDTEMPPREFIX must be the name of an interface that was previously defined by an INTERFACE statement.

prefix/prefix_length

The digits (in colon-hexadecimal format) before the slash (/) represent the prefix. The *prefix_length* value represents the length of the prefix in bits. Valid values for *prefix_length* parameter are in the range 1 - 64.

ALL Causes temporary addresses to be generated for all prefixes that are learned over this interface by way of router advertisements.

DELTEMPPREFIX

Use the DELTEMPPREFIX keyword to delete prefixes from the temporary prefixes list of an existing INTERFACE definition. The temporary prefixes list limits the set of prefixes for which temporary IPv6 addresses can be generated. A temporary IPv6 address is generated when a router advertisement containing the prefix is processed and the prefix is included in one of the prefixes in the temporary prefixes list. The *intf_name* variable coded with the DELTEMPPREFIX keyword must be the name of an interface that was previously defined by an INTERFACE statement.

prefix/prefix_length

The digits (in colon-hexadecimal format) before the slash (/) represent the prefix. The *prefix_length* value represents the length of the prefix in bits. Valid values for the *prefix_length* are in the range 1 - 64. All temporary addresses for this interface whose prefix is not included in the updated temporary prefixes list are deleted.

ALL Delete all prefixes from the temporary prefixes list, which sets the temporary prefixes list to NONE. All temporary addresses for this interface are deleted, and no more temporary addresses are generated for this interface.

IPADDR *ipaddr_spec*

TCP/IP always creates the link-local IPv6 address. If IPADDR is not specified, then TCP/IP enables autoconfiguration for the interface.

Tip: Autoconfiguration is enabled if there is a router or some other device that provides a router advertisement.

If no address or prefix is specified, it is obtained from a router on the LAN by way of an IPv6 stateless autoconfiguration. For more information, see z/OS Communications Server: IPv6 Network and Application Design Guide.

IPAQENET6
Indicates that the interface uses the interface based on IP assist, belongs to the QDIO family of interfaces, and uses the Gigabit Ethernet or Fast Ethernet protocol.

INTFID *interface_id*
An optional 64-bit interface identifier in colon-hexadecimal format. IPv6 shorthand is not allowed when specifying the interface ID. If specified, this interface ID is used to form the link-local address for the interface, and is also appended to any manually configured prefixes for the interface, to form complete IPv6 addresses on the interface. If you do not configure manual IP addresses on the interface, the INTFID value is appended to any prefixes that are learned over this interface by way of router advertisements to form public IPv6 addresses on the interface. The INTFID value is not used to form temporary IPv6 addresses. A randomly generated interface ID is appended to any learned prefixes to form temporary IPv6 addresses on the interface (if temporary addresses are enabled).

If INTFID is not coded, TCP/IP builds the Interface ID using information returned from the OSA-Express Adapter (during Interface activation). The built Interface ID value is then used to form the link-local address. This value is also used to complete the formation of other IPv6 addresses on the interface, if you choose to configure only the prefix portion of the addresses (by way of IPADDR or ADDADDR). Also, if you do not configure manual IP addresses on the interface, the built interface ID value is appended to any prefixes learned over this interface by way of router advertisements to form public IPv6 addresses on the interface. The built interface ID value is not used to form temporary IPv6 addresses. A randomly generated interface ID is appended to any learned prefixes to form temporary IPv6 addresses on the interface (if temporary addresses are enabled).

When defining the interface ID, the local/universal flag (the U bit, bit 6 shown in the following example) must be set to 0. The group/individual flag (the G bit, bit 7 shown in the following example) must also be set to 0. If either flag is set incorrectly, interface definition fails. Additionally, an interface ID value correlating to an ISATAP address or a Reserved Anycast address is not allowed. (An ISATAP Interface ID has '00005EFE'x in bits 0 - 31, and a Reserved Anycast Interface ID has 'FCFFFFFFFFFFFF8' in bits 0 - 56.)

```
|              1|1              3|3              4|4              6|
|0             5|6              1|2              7|8              3|
+----------------+----------------+----------------+----------------+
|xxxxxxUGxxxxxxxx|xxxxxxxxxxxxxxxx|xxxxxxxxxxxxxxxx|xxxxxxxxxxxxxxxx|
+----------------+----------------+----------------+----------------+
```

SOURCEVIPAINTERFACE *vipa_name*
SOURCEVIPAINTERFACE is optional. Use this parameter to specify which previously defined static VIPA interface is to be used for SOURCEVIPA (when IPCONFIG6 SOURCEVIPA is in effect).

Tip: The use of the SOURCEVIPAINTERFACE parameter can be overridden. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

The *vipa_name* is the interface name for a VIRTUAL6 interface. If the VIPA has multiple IP addresses, then the sourcevipa address for outbound packets is selected from among these addresses according to the default source address selection algorithm. For more information, seez/OS Communications Server: IPv6 Network and Application Design Guide.

Requirement: The VIRTUAL6 interface must be defined prior to specifying this INTERFACE statement to the TCP/IP stack. It must either already be defined or, the INTERFACE statement that defines it must precede this INTERFACE statement in the profile data set.

CHPID *chpid*
This parameter applies only to interfaces of CHPIDTYPE OSX and is used to specify the CHPID for the interface. This value is a 2-character hexadecimal value (00 - FF).

PORTNAME *portname*
Use this parameter to specify the PORT name contained in the TRLE definition for the QDIO interface. The TRLE must be defined as MPCLEVEL=QDIO. For details about defining a TRLE, see z/OS Communications Server: SNA Resource Definition Reference.

Requirement: The *portname* value must be different from the name that is specified for the *intf_name* value.

NONROUTER
If a datagram is received at this interface for an unknown IP address, the datagram is not routed to this TCP/IP instance. This is the default value.

PRIRouter and SECRouter parameters interact with the VLANID parameter. See the VLANID parameter to understand this relationship.

For more information about VLANID parameter interactions, see z/OS Communications Server: IP Configuration Guide.

Rule: This keyword applies only to interfaces of CHPIDTYPE OSD and is ignored if the VMAC parameter is configured on the INTERFACE statement.

PRIROUTER
If a datagram is received at this interface for an unknown IP address and is not destined for a virtual MAC, the datagram is routed to this TCP/IP instance.

Rule: This keyword applies only to interfaces of CHPIDTYPE OSD and is ignored if the VMAC parameter is configured on the INTERFACE statement.

SECROUTER
If a datagram is received at this interface for an unknown IP address and is not destined for a virtual MAC, and there is no active TCP/IP instance defined as PRIROUTER, then the datagram is routed to this TCP/IP instance.

Rule: This keyword applies only to interfaces of CHPIDTYPE OSD and is ignored if the VMAC parameter is configured on the INTERFACE statement.

DUPADDRDET *count*
Use this parameter to specify the number of times to attempt duplicate address detection. The minimum value is 0, maximum is 2 and default is 1. This is an optional parameter.

Guideline: A value of 0 means that TCP/IP does not perform duplicate address detection for this interface.

MTU *num*
The maximum transmission unit (MTU) in bytes. This value can be up to 9000. The minimum MTU for IPv6 is 1280. The stack takes the minimum of the configured value and the value supported by the device (returned by the OSA adapter).

The MTU default, which depends on value supported by device, is the following value:

- Gigabit Ethernet default MTU = 9000
- Fast Ethernet default MTU = 1500

Tip: See z/OS Communications Server: IP Configuration Guide, in section Maximum transmission unit considerations, for additional information about how TCP/IP uses the MTU to determine the largest size frame to send.

VLANID id

Specifies the decimal virtual LAN identifier to be assigned to the OSA-Express INTERFACE. This field should be a virtual LAN identifier recognized by the switch for the LAN connected to this OSA-Express. The valid range is 1 - 4094. This parameter is optional for CHPIDTYPE OSD and required for CHPIDTYPE OSX.

Guideline: Installation configuration on other platforms or related to Ensemble networking can limit the maximum VLANID of 4096.

The VLANID parameter interacts with the PRIRouter and SECRouter parameters. If you configure both the VLANID parameter and either PRIROUTER or SECROUTER parameter, then this TCP/IP instance acts as a router for this VLAN (ID) only. Datagrams that are received at this device instance for an unknown IP address and are not destined for a virtual MAC are routed only to this TCP/IP instance if it is VLAN tagged with this VLAN ID. For more information about VLANID parameter interactions, see z/OS Communications Server: IP Configuration Guide.

Rule: If you are configuring multiple VLAN interfaces to the same OSA-Express feature, then you must specify the VMAC parameter (with the default ROUTEALL attribute) on the INTERFACE statement for each of these interfaces.

Restriction: The stack supports a maximum of 32 IPv6 VLAN interfaces to the
| same OSA-Express port. Additional VLANID limitations might exist if this
| interface can be used with Shared Memory Communications over Remote
| Direct Memory Access (SMC-R). See VLANID considerations in z/OS
| Communications Server: IP Configuration Guide for details.

READSTORAGE

An optional parameter indicating the amount of fixed storage that z/OS Communications Server should keep available for read processing for this adapter. The QDIOSTG VTAM start option allows you to specify a value which applies to all OSA-Express adapters in QDIO mode. You can use the READSTORAGE keyword to override the global QDIOSTG value for this adapter based on the inbound workload you expect over this interface on this stack. The valid values are:

GLOBAL
The amount of storage is determined by the QDIOSTG VTAM start option. This is the default value.

MAX Use this value if you expect a heavy inbound workload over this interface.

AVG Use this value if you expect a medium inbound workload over this interface.

MIN Use this value if you expect a light inbound workload over this interface.

Tip: See the description of the QDIOSTG VTAM start option in the z/OS Communications Server: SNA Resource Definition Reference for details about exactly how much storage is allocated by z/OS Communications Server for each of these values.

Rule: If you define both a LINK and INTERFACE statement for the same adapter, then the READSTORAGE value on the LINK statement must match the READSTORAGE value on the corresponding INTERFACE statement. If you define an INTERFACE statement that contains a value for READSTORAGE that conflicts with the READSTORAGE value for a previous LINK statement for the same adapter, then TCP/IP rejects the INTERFACE statement.

INBPERF

An optional parameter that indicates how processing of inbound traffic for the QDIO interface is performed.

There are three supported static settings (MINCPU, MINLATENCY, and BALANCED).that indicate how frequently the adapter should interrupt the host for inbound traffic: BALANCED, MINCPU, and MINLATENCY. The static settings use static interrupt-timing values. The static values are not always optimal for all workload types or traffic patterns, and cannot account for changes in traffic patterns.

There is also one supported dynamic setting (DYNAMIC). This setting causes the host (stack) to dynamically adjust the timer-interrupt value while the device is active and in use. This function exploits an OSA hardware function called Dynamic LAN Idle. Unlike the static settings, the DYNAMIC setting reacts to changes in traffic patterns, and sets the interrupt-timing values at the point where throughput is maximized. In addition, the DYNAMIC setting uses the OSA Dynamic Router Architecture function to enable QDIO inbound workload queues for specific inbound traffic types.

Result: When you specify OLM on the INTERFACE statement, the INBPERF parameter is ignored and the statement defaults to the value DYNAMIC.

Valid values are:

BALANCED

This setting uses a static interrupt-timing value, which is selected to achieve reasonably high throughput and reasonably low CPU consumption. This is the default value for CHPIDTYPE OSD.

DYNAMIC

This setting causes the host to dynamically signal the OSA-Express feature to change the timer-interrupt value, based on current inbound workload conditions. The DYNAMIC setting is effective only for OSA-Express2 or later features on at least an IBM System z9 that supports the corresponding Dynamic LAN Idle function. See the 2094DEVICE Preventive Service Planning (PSP) bucket and the 2096DEVICE Preventive Service Planning (PSP) bucket for more information about the level of OSA-Express2 adapter that supports this function. See the 2097DEVICE Preventive Service Planning (PSP) bucket for more information about the OSA-Express3 adapter that supports this function. The DYNAMIC setting can decrease latency and provide increases in throughput for many interactive workloads. For all other workload combinations, this setting provides performance similar to the three static settings. This is the default value for CHPIDTYPE OSX.

If the DYNAMIC setting is specified for an OSA-Express adapter that does not support the dynamic LAN Idle function, the stack reverts to using the BALANCED setting.

WORKLOADQ | NOWORKLOADQ

This subparameter controls the QDIO inbound workload queueing function for the interface. QDIO inbound workload queueing is effective only for OSA-Express features in QDIO mode that support the corresponding Data Router Architecture. OSA-Express features that support workload queueing do not necessarily support workload queueing for all possible traffic types. For more information about the QDIO inbound workload queueing function and the OSA-Express features that support it, see QDIO inbound workload queueing in z/OS Communications Server: IP Configuration Guide.

NOWORKLOADQ

Specifies that QDIO inbound workload queueing is not enabled for inbound traffic. All inbound traffic for this interface uses a single input queue. This is the default.

WORKLOADQ

Specifies that QDIO inbound workload queueing is enabled for inbound traffic.

If the WORKLOADQ subparameter is specified, QDIO inbound workload queueing is enabled for specific inbound traffic types. A primary input queue is reserved for all other traffic types.

Ancillary input queues (AIQs) are created for the following inbound traffic types when supported by the OSA-Express feature:

- Sysplex distributor
- Streaming workloads (for example FTP)
- Enterprise Extender (EE)

Requirement: You must specify the VMAC parameter with WORKLOADQ to enable QDIO inbound workload queueing.

If the WORKLOADQ setting is specified for an OSA-Express adapter that does not support the Data Router Architecture function, the stack reverts to using a single input queue.

MINCPU

This setting uses a static interrupt-timing value, which is selected to minimize host interrupts without regard to throughput. This mode of operation might result in minor queueing delays (latency) for packets into the host, which is not optimal for workloads with demanding latency requirements.

MINLATENCY

This setting uses a static interrupt-timing value, which is selected to minimize latency (delay), by more aggressively presenting received packets to the host. This mode of operation generally results in higher CPU consumption than the other three settings. Use this setting only if host CPU consumption is not an issue.

Rule: If you define both a LINK IPAQENET and an INTERFACE IPAQENET6 statement for the same adapter, then the following rules apply for the INBPERF parameter on these statements:

- The value on the LINK statement must match the INBPERF value on the corresponding INTERFACE statement.
- The INTERFACE statement supports the subparameters WORKLOADQ and NOWORKLOADQ for the INBPERF DYNAMIC parameter. These subparameters are associated with QDIO inbound workload queueing support and are not supported on the LINK IPAQENET statement. So, if you specify the INBPERF DYNAMIC parameter for both the LINK and the INTERFACE statements, then you must use the default or specify the NOWORKLOADQ subparameter for the INBPERF DYNAMIC parameter on the INTERFACE statement. This ensures that the INBPERF DYNAMIC setting for both statements is the same.
- If you define an INTERFACE IPAQENET6 statement that contains a value for INBPERF that conflicts with the INBPERF value for a previous LINK IPAQENET statement for the same adapter, then TCP/IP rejects the INTERFACE statement.

SECCLASS *security_class*

Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC6RULE statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG6 statement.

MONSYSPLEX | NOMONSYSPLEX

Specifies whether or not sysplex autonomics should monitor the interface's status.

NOMONSYSPLEX

Specifies that sysplex autonomics should not monitor the interfaces's status. This is the default value.

MONSYSPLEX

Specifies that sysplex autonomics should monitor the interface's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this interface is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

DYNVLANREG | NODYNVLANREG

This parameter controls whether or not the VLAN ID for this interface is dynamically or statically registered with the physical switch on the LAN.

Restriction: This parameter is applicable only if a VLAN ID is specified on the statement.

Dynamic registration of VLAN IDs is handled by the OSA-Express feature and the physical switch on your LAN. Therefore, in order for the DYNVLANREG parameter to be effective, both must be at a level which provides the necessary hardware support for dynamic VLAN ID registration. After the interface is active, you can view the Netstat DEvlinks/**-d** report output to determine if your OSA-Express feature can support VLAN dynamic registration. This Netstat report also displays whether or not dynamic VLAN ID registration has been configured for the interface.

Rule: If you define both a LINK and INTERFACE statement for the same adapter, then the dynamic VLAN ID registration parameter value on the LINK statement must match the value of this same parameter on the corresponding INTERFACE statement. If you define an INTERFACE statement that contains a dynamic VLAN ID registration parameter value that conflicts with the same parameter value for a previous INTERFACE statement for the same OSA-Express feature, then TCP/IP rejects the INTERFACE statement.

NODYNVLANREG
Specifies that if a VLAN ID is configured for this interface, it must be manually registered with the physical switches on the corresponding LAN. This is the default value. If this parameter is specified without a VLAN ID, then it is ignored.

DYNVLANREG
Specifies that if a VLAN ID is configured for this interface, it is dynamically registered with the physical switches on the corresponding LAN. If this parameter is specified without a VLAN ID, then warning message EZZ0056I is issued and the NODYNVLANREG setting is used instead.

VMAC *macaddr*
Specifies the virtual MAC address, which can be represented by 12 hexadecimal characters. The OSA-Express device uses this address rather than the physical MAC address of the device for all IPv6 packets to and from this TCP/IP stack. For CHPIDTYPE OSD, using a virtual MAC address is optional. For CHPIDTYPE OSX, using a virtual MAC address is required, so the VMAC parameter is the default.

The *macaddr* value is optional for CHPIDTYPE OSD and cannot be specified for CHPIDTYPE OSX. If the *macaddr* value is not coded, then the OSA-Express device generates a virtual MAC address. If the *macaddr* is coded, it must be defined as a locally administered individual MAC address. This means the MAC address must have bit 6 (the universal or local flag U bit) of the first byte set to 1 and bit 7 (the group or individual flag G bit) of the first byte set to 0. The second hexadecimal character must be 2, 6, A or E. The bit positions within the 12 hexadecimal characters are indicated as follows:

```
|               1|1              3|3              4|
|0              5|6              1|2              7|
+----------------+----------------+----------------+
|xxxxxxUGxxxxxxxx|xxxxxxxxxxxxxxxx|xxxxxxxxxxxxxxxx|
+----------------+----------------+----------------+
```

Rules:

- The same virtual MAC address generated by the OSA-Express device at interface activation remains in effect for this OSA-Express for this TCP/IP stack, even if the interface is stopped or becomes inoperative (INOPs). A new Virtual MAC address is generated only if the INTERFACE statement is deleted and redefined, or if the TCP/IP stack is recycled.

- The NONROUTER, PRIROUTER, and SECROUTER parameters are ignored for an OSA-Express interface if the VMAC parameter is configured on the INTERFACE statement.

Guideline: Unless the virtual MAC address representing this OSA-Express device must remain the same even after TCP/IP termination and restart, configure VMAC without a *macaddr* value and allow the OSA-Express device to generate it. This guarantees that the VMAC address is unique from all other physical burned-in MAC addresses and from all other VMAC addresses generated by any OSA-Express feature.

ROUTEALL
Specifies that all IP traffic destined to the virtual MAC is forwarded by the OSA-Express device to the TCP/IP stack. This is the default value. See the router information in z/OS Communications Server: IP Configuration Guide for more details.

ROUTELCL
This specifies that only traffic destined to the virtual MAC and whose destination IP address is registered with the OSA-Express device by this TCP/IP stack is forwarded by the OSA-Express. See the router information in z/OS Communications Server: IP Configuration Guide for more details.

SMCR | NOSMCR
Specifies whether this interface can be used with Shared Memory Communications over Remote Direct Memory Access (SMC-R) for external data network communications.

NOSMCR
Specifies that this interface cannot be used for new TCP connections with SMC-R for external data network communications.

SMCR
Specifies that this interface can be used for new TCP connections with SMC-R for external data network communications. This is the default setting.

Rules:
- SMCR and NOSCMR are valid with CHPIDTYPE OSD definitions only.
- SMCR has no effect unless at least one Peripheral Component Interconnect Express (PCIe) function ID (PFID) value is specified by using the PFID subparameter of the SMCR parameter on the GLOBALCONFIG statement.

OLM| NOOLM
An optional parameter indicating whether an OSA-Express adapter operates in optimized latency mode.

OLM Specifies that the OSA-Express adapter operates in optimized latency mode (OLM). Optimized latency mode optimizes interrupt processing for both inbound and outbound data. Use this mode for workloads that have demanding latency requirements. Because this mode can provide significant increases of throughput, this mode is particularly suited for interactive, non-streaming workloads. For more information about OLM, see the optimized latency mode topic in z/OS Communications Server: IP Configuration Guide.

NOOLM
Specifies that the OSA-Express adapter should not operate in optimized latency mode. This is the default value.

Guidelines:

- Because of the operating characteristics of optimized latency mode, you might need to change configuration to direct traffic to particular OSA-Express write priority queues and to limit the number of concurrent users sharing an OSA-Express adapter configured for OLM. See the optimized latency mode topic in z/OS Communications Server: IP Configuration Guide. for more information.
- The optimized latency mode function targets a z/OS environment with high-volume interactive workloads. Although optimized latency mode can compensate for some mixing of workloads, an excessive amount of high-volume streaming workloads, such as bulk data or file transfer, can result in higher CPU consumption.

Restrictions:

- This function is limited to OSA-Express3 or later Ethernet features in QDIO mode that are running with an IBM System z10 or later. See the 2097 DEVICE Preventive Service Planning (PSP) bucket for more information.
- For an OSA-Express configured to use optimized latency mode, the stack ignores the configured or default INBPERF setting and uses the value DYNAMIC.

NOISOLATE | ISOLATE

Specifies whether packets should be directly routed between TCP/IP stacks that share the OSA adapter.

NOISOLATE

Route packets directly between TCP/IP stacks that share the OSA adapter. In this mode, if the next hop address was registered by another stack that is sharing the OSA, then OSA-Express routes the packet directly to the sharing stack without putting the packet on the external LAN.

ISOLATE

Prevent OSA-Express from routing packets directly to another TCP/IP stack that is sharing the OSA adapter. In this mode, OSA-Express discards any packets when the next hop address was registered by another stack that is sharing the OSA adapter. In this mode, packets can flow between two stacks that share the OSA adapter only by first going through a router on the LAN. For more details, see OSA-Express connection isolation information in z/OS Communications Server: IP Configuration Guide.

Tips:

- If you isolate an INTERFACE, that action might have an adverse effect on latency.
- You can selectively apply OSA-Express connection isolation to individual virtual LANs.
- OSA-Express requires that both stacks sharing the port be non-isolated for direct routing to occur. Therefore, for traffic between two stacks sharing the OSA adapter, as long as at least one of the stacks is isolated, connection isolation is in effect for traffic in both directions between these stacks.

Restriction: This function is limited to OSA-Express2 or later Ethernet features in QDIO mode and running at least an IBM System z9 Enterprise Class (EC) or z9 Business Class (BC). See the 2094, 2096, 2097, or 2098 DEVICE Preventive Service Planning (PSP) and the 2096DEVICE Preventive Service Planning (PSP) buckets for more information.

TEMPPREFIX

TEMPPREFIX specifies the set of prefixes for which temporary IPv6 addresses can be generated. A temporary IPv6 address is generated when a router advertisement containing a prefix is processed and the prefix is included in one of the prefixes in the temporary prefix list. For example, if TEMPPREFIX 2001:0db8:58cd::/48 is specified for an interface, a temporary address is generated for advertised prefix 2001:0db8:58cd:0001/64; however, a temporary address is not generated for advertised prefix 2001:0db8:5555:0001/64.

ALL Generate temporary addresses for all prefixes that are learned over this interface by way of router advertisements. ALL is the default.

NONE
No IPv6 temporary addresses are generated for this interface.

prefix/prefix_length
The digits (in colon-hexadecimal format) before the slash (/) represent the prefix. The *prefix_length* value represents the length of the prefix, in bits. Valid values for *prefix_length* are in the range 1 - 64.

Rules:

- Temporary addresses are generated only on an interface that is enabled for stateless address autoconfiguration.
- Temporary addresses are generated only when the TEMPADDRS keyword is specified on the IPCONFIG6 statement.

Requirement: You must specify the job name of an application in the SRCIP statement block with a value of TEMPADDRS to cause a temporary IPv6 address to be preferred over a public IPv6 address as the source IP address for the application; otherwise, the default source address selection algorithm prefers public IPv6 addresses over temporary addresses. For more information, see the information about the default source address selection algorithm in z/OS Communications Server: IPv6 Network and Application Design Guide.

Steps for modifying

See "Summary of INTERFACE statements" on page 141 for modification information.

Examples

```
INTERFACE OSAQDIO26 ; OSA QDIO (Fast Ethernet)
DEFINE IPAQENET6
PORTNAME OSAQDIO2
SOURCEVIPAINT VIPAV6
IPADDR 2001:0DB8:1:9:67:115:66     ; (Global Address)
```

Usage notes

Restriction: For each interface, the PRIROUTER and SECROUTER attributes can be in effect for only one TCP/IP instance within a central processor complex (CPC). If PRIROUTER is specified for an IPAQENET6 interface, but the IPv6 primary router attribute is already in effect on another TCP/IP instance for the same OSA-Express, then TCP/IP issues a warning message during interface activation and ignores the PRIROUTER parameter. Therefore, only one TCP/IP instance can be the primary router for the OSA-Express. Depending on the level of OSA-Express being started, either only one or multiple TCP/IP instances can be allowed to have SECROUTER specified. If OSA-Express allows only one secondary router, any TCP/IP instance subsequently starting that interface with SECROUTER receives a warning message during START processing for the interface. If OSA-Express allows multiple secondary routers, then OSA-Express can select any TCP/IP instance which specifies SECROUTER as the secondary router. There is no requirement that the same TCP/IP instance be specified PRIROUTER or SECROUTER for all OSA-Express adapters attached to the CPC.

Rule: To configure a single OSA port for both IPv4 and IPv6 traffic, consider the following conditions:

- If you use DEVICE/LINK/HOME for the IPv4 definition and INTERFACE for the IPv6 definition, the PORTNAME value on the INTERFACE statement must match the device_name on the DEVICE statement. This combination shares a single DATAPATH device.
- If you use INTERFACE for both IPv4 and IPv6 definitions, the PORTNAME value on the IPv4 INTERFACE statement must match the PORTNAME value on the IPv6 INTERFACE statement. This combination results in separate DATAPATH devices.

Related topics

- “BEGINROUTES statement” on page 28
- “DEVICE and LINK — MPCIPA OSA-Express QDIO devices statement” on page 74
- “GLOBALCONFIG statement” on page 117
- “INTERFACE - IPAQENET OSA-Express QDIO interfaces statement” on page 145
- “START statement” on page 292
- “STOP statement” on page 293

INTERFACE — IPAQIDIO6 HiperSockets interfaces statement

Use the INTERFACE statement for IPAQIDIO6 to configure IPv6 HiperSockets connectivity. Use the CHPID parameter to specify the value of the desired IQD CHPID that was configured within HCD. HiperSockets interfaces do not require a corresponding TRLE definition. Instead, the TRLE is dynamically built when the interface is started.

Rule: Specify the required parameters in the order shown here. The Interface Definition parameters can be specified in any order.

Syntax

```
►►──INTERFace──intf_name──┬─DEFINE──IPAQIDIO6──┤ Interface Definition ├─┬──────────►◄
                          ├─DELEte──────────────────────────────────────┤
                          │           ┌──────────────┐                  │
                          │           ▼              │                  │
                          ├─ADDADDR────ipaddr_spec───┴───┤              │
                          │           ┌──────────────┐   │              │
                          │           ▼              │   │              │
                          ├─DELADDR────ipaddr_spec───┴───┤              │
                          │            ┌──────────────┐  │              │
                          │            ▼              │  │              │
                          └─DEPRADDR────ipaddr_spec───┴──┘──────────────┘
```

Interface Definition:

```
├──CHPID ──chpid──┬─────────────────────────┬──────────────────────────────────────►
                  └─INTFID ──interface_id ──┘

►──┬────────────────────────────────────┬──┬─READSTORAGE GLOBAL──────┬──┬─────────────┬──►
   │           ┌─────────────────┐      │  └─READSTORAGE──┬─MAX─┬────┘  └─VLANID id───┘
   │           ▼                 │      │                 ├─AVG─┤
   └─IPADDR ────┤ ipaddr_spec ├──┴──────┘                 └─MIN─┘

►──┬─────────────────────────────────┬──┬─SECCLASS 255─────────────┬──────────────────►
   └─SOURCEVIPAINTerface ──vipa_name ─┘  └─SECCLASS security_class──┘

►──┬─NOMONSYSPLEX─┬──────────────────────────────────────────────────────────────────┤
   └─MONSYSPLEX───┘
```

ipaddr_spec:

```
├──┬─ipv6_address──────────┬───────────────────────────────────────────────────────┤
   └─prefix/prefix_length──┘
```

Parameters

intf_name
: The name of the interface. The maximum length is 16 characters.

  Restriction: Do not specify the value PUBLICADDRS or TEMPADDRS for the interface name. The values PUBLICADDRS and TEMPADDRS are keywords on the SRCIP statement. These values are not recognized if they are specified as an IPv6 interface name on an SRCIP entry.

DEFINE
: Specifies that this definition is to be added to the list of defined interfaces.

DELETE
: Specifies that this definition is to be deleted from the list of defined interfaces. The *intf_name* must be the name of an interface previously defined by an INTERFACE statement. INTERFACE DELETE deletes all home IP addresses for the interface.

ADDADDR *ipaddr_spec*
Adds IP addresses to an existing INTERFACE definition (similar to an obeyfile to update the home list) without having to delete and redefine the INTERFACE. The interface name (*intf_name*) coded with ADDADDR must be the name of an interface previously defined by an INTERFACE statement.

DELADDR *ipaddr_spec*
Deletes IP addresses from an existing INTERFACE definition. The DELADDR parameter is valid only for an IP address or prefix configured manually. The interface name (*intf_name*) coded with DELADDR must be the name of an interface previously defined by an INTERFACE statement. DELADDR is valid only in a VARY OBEYFILE profile.

Guideline: If you specify a prefix for DELADDR, then the only IP addresses affected are those defined by way of the same prefix specified on IPADDR or ADDADDR.

DEPRADDR *ipaddr_spec*
The DEPRADDR keyword allows you to deprecate an IP address. This can assist with site renumbering. DEPRADDR is valid only for an IP address or prefix configured manually. If you use DEPRADDR to deprecate an IP address, you can subsequently use ADDADDR again to make that IP address preferred. For DEPRADDR, the *interface_name* must be the name of an interface previously defined by an INTERFACE statement. DEPRADDR is valid only in a VARY OBEYFILE profile.

Guideline: If you specify a prefix for DEPRADDR, then the only IP addresses affected are those defined by way of the same prefix specified on IPADDR or ADDADDR.

IPADDR *ipaddr_spec*
The IPADDR parameter is optional, and is used to configure the interface's IPv6 addresses other than the link-local address (which is generated internally by TCP/IP).

Rule: Stateless Address Autoconfiguration does not apply to IPAQIDIO6 interfaces, you must manually configure any addresses (other than link-local) that are to be assigned to the IPAQIDIO6 interface.

If ADDADDR, DELADDR, DEPRADDR, or IPADDR is specified, then *ipaddr_spec* can be one of the following:

- ipv6_addr (A fully qualified IPv6 address in colon-hexadecimal format)
- prefix/prefix_length. Here, the digits (in colon-hexadecimal format) before the / represent the prefix. The prefix length represents the length of the prefix in bits. If a prefix length is coded, it must be equal to 64. When a prefix is specified, TCP/IP forms the IPv6 address by appending an interface ID to the specified prefix. The selected interface ID is either the value specified by way of the INTFID keyword, or the value returned by the device when the interface was started.

For information about the IPv6 address restrictions, see "INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement" on page 161.

IPAQIDIO6
Indicates that the interface is for HiperSockets IPv6.

INTFID *interface_id*
An optional 64-bit interface identifier in colon-hexadecimal format.

If specified, this interface ID is used to form the link-local address for the interface, and is also appended to any manually configured prefixes for the interface, to form complete IPv6 addresses on the interface.

If INTFID is not coded, TCP/IP builds the Interface ID using information returned from the HiperSockets device (during interface activation). The built Interface ID value is then used to form the link-local address. This value is also used to complete the formation of other IPv6 addresses on the interface, if you choose to configure only the prefix portion of the addresses (by way of IPADDR or ADDADDR).

For information about INTFID restrictions, see “INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement” on page 161.

SOURCEVIPAINTERFACE *vipa_name*
SOURCEVIPAINTERFACE is optional. Use this to specify which previously defined VIPA interface is to be used for SOURCEVIPA (when IPCONFIG6 SOURCEVIPA is in effect).

Tip: The use of the SOURCEVIPAINTERFACE parameter can be overridden. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.The *vipa_name* is the interface name for a VIRTUAL6 interface. If the VIPA has multiple IP addresses, then the sourcevipa address for outbound packets is selected from among these addresses according to the default source address selection algorithm. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.

Requirement: The VIRTUAL6 interface must be defined prior to specifying this INTERFACE statement to the TCP/IP stack. It must either already be defined or, the INTERFACE statement that defines it must precede this INTERFACE statement in the profile data set.

SECCLASS *security_class*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC6RULE statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if IPSECURITY is not specified on the IPCONFIG6 statement.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the interface's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the interface's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor tinterface's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over the interface is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

The following interface-specific values can be specified for IPAQIDIO6.

CHPID *chpid*

Use this parameter to specify the IQD CHPID for the HiperSockets interface. This value is a 2-character hexadecimal value (00x - FFx). The hexadecimal value specified on the CHPID parameter cannot be the same value that is used for the dynamic XCF HiperSockets interface. See IQDCHPID start option in the z/OS Communications Server: SNA Resource Definition Reference.

READSTORAGE

An optional parameter indicating the amount of fixed storage that z/OS CS should keep available for read processing for this interface. The IQDIOSTG VTAM start option allows you to specify a value which applies to all HiperSockets devices. You can use the READSTORAGE keyword to override the global IQDIOSTG value for this interface based on the inbound workload you expect over this interface on this stack. The valid values are:

GLOBAL

The amount of storage is determined by the IQDIOSTG VTAM start option. This is the default value.

MAX Use this value if you expect a heavy inbound workload over this interface.

AVG Use this value if you expect a medium inbound workload over this interface.

MIN Use this value if you expect a light inbound workload over this interface.

Tip: See the description of IQDIOSTG start option in the z/OS Communications Server: SNA Resource Definition Reference for details about exactly how much storage is allocated by z/OS Communications Server for each of these values.

Rules:

- If you define both a LINK and INTERFACE statement for the same device, then the READSTORAGE value on the LINK statement must match the READSTORAGE value on the corresponding INTERFACE statement.
- If you define an INTERFACE statement which contains a value for READSTORAGE which conflicts with the READSTORAGE value for a previous LINK statement for the same device, then TCP/IP rejects the INTERFACE statement.

VLANID *id*

An optional parameter followed by a decimal number indicating the virtual LAN identifier to be assigned to this HiperSockets interface. The valid range is 1 - 4094.

Restriction: HiperSockets allows a stack to specify only one VLAN ID if you define IPv4 connectivity by using DEVICE and LINK statements and also configure an IPv6 INTERFACE statement for the same CHPID. In this case, if you specify a different VLAN ID value on a LINK or INTERFACE definition for the same CHPID, the second statement is rejected.

Steps for modifying

See "Summary of INTERFACE statements" on page 141 for modification information.

Examples

```
INTERFACE HIPERSOCK1 DEFINE IPAQIDIO6 CHPID FC
            IPADDR 12AB::7
```

Usage notes

Rule: To configure a single HiperSockets CHPID for both IPv4 and IPv6 traffic, consider the following conditions:

- If you use DEVICE/LINK/HOME for the IPv4 definition and INTERFACE for the IPv6 definition, the CHPID value on the INTERFACE statement must match the xx portion of the device_name (IUTIQDxx) on the DEVICE statement. This combination shares a single DATAPATH device.
- If you use INTERFACE for both IPv4 and IPv6 definition, the CHPID value on the IPv4 INTERFACE statement must match the CHPID value on the IPv6 INTERFACE statement. This combination results in separate DATAPATH devices.

Related topics

- "BEGINROUTES statement" on page 28
- "DEVICE and LINK — MPCIPA HiperSockets devices statement" on page 85
- "INTERFACE — IPAQIDIO HiperSockets interfaces statement" on page 157
- "START statement" on page 292
- "STOP statement" on page 293

INTERFACE — LOOPBACK6 interface statement

There is only one LOOPBACK6 interface. The default LOOPBACK6 address ::1 is generated automatically and cannot be deleted. Therefore, you cannot DEFINE or DELETE the LOOPBACK6 interface. However, you can add additional IP addresses for LOOPBACK6 in the initial profile or by using the VARY TCPIP,,OBEYFILE command. Additionally, you can delete and deprecate one or more of these additional IP addresses by using the VARY TCPIP,,OBEYFILE command.

Rule: Specify the required parameters in the order shown here. The optional parameters can be specified in any order.

Syntax

```
                                      ┌──────────────┐
                                      ▼              │
►►──INTERFace──LOOPBACK6──┬─ADDADDR───ipaddr_spec────┴─┬──────────────────►◄
                          │           ┌──────────────┐ │
                          │           ▼              │ │
                          ├─DELADDR───ipaddr_spec────┴─┤
                          │            ┌──────────────┐│
                          │            ▼              ││
                          └─DEPRADDR───ipaddr_spec────┴┘
```

Parameters

ADDADDR
: Allows the addition of IP addresses to an existing LOOPBACK6 definition (similar to updating the HOME list with a VARY TCPIP,,OBEYFILE command) without having to delete and redefine the INTERFACE.

  If ADDADDR is specified, then *ipaddr_spec* can be one or more full IPv6 addresses.

ipaddr_spec
: For information about the IPv6 address restrictions, see "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143.

  The following value can be specified for ipaddr_spec:

  ipv6_address
  : IPv6 address in colon-hexadecimal format.

DELADDR *ipaddr_spec*
: Allows you to delete IP addresses from an existing LOOPBACK6 definition.

  If DELADDR is specified, then *ipaddr_spec* can be one or more full IPv6 addresses.

  Restriction: You cannot code DELADDR to delete the default LOOPBACK6 address ::1.

DEPRADDR *ipaddr_spec*
: The DEPRADDR keyword allows you to deprecate an IP address. This can assist with site renumbering. If you use DEPRADDR to deprecate an IP address, you can subsequently use ADDADDR again to make that IP address preferred.

  DEPRADDR is valid only for an IP address configured manually.

Examples

```
INTERFACE  LOOPBACK6  ADDADDR  ::0014:0
```

INTERFACE — MPCPTP6 interfaces statement

The MPC Point-To-Point Data Link Control supports IPv6 traffic. With this support, interface type MPCPTP6 can be used to carry IPv6 traffic over ESCON channels, over XCF links in a sysplex, or between z/OS Communications Server images using the simulated device provided by the IUTSAMEH function in VTAM.

Rule: Specify the required parameters in the order shown here. The Interface Definition parameters can be specified in any order.

Syntax

```
►►──INTERFace──intf_name──┬─DEFINE──MPCPTP6──┤ Interface Definition ├─┬───────────►◄
                          ├─DELETE──────────────────────────────────┤
                          │           ┌──────────────┐              │
                          │           ▼              │              │
                          ├─ADDADDR─────ipaddr_spec──┴──────────────┤
                          │           ┌──────────────┐              │
                          │           ▼              │              │
                          ├─DELADDR─────ipaddr_spec──┴──────────────┤
                          │            ┌──────────────┐             │
                          │            ▼              │             │
                          └─DEPRADDR─────ipaddr_spec──┴─────────────┘
```

Interface Definition:

```
                                                                ┌─NOMONSYSPLEX─┐
├──┬─────────────────────────┬──┬───────────────────────────────┬──┼──────────────┼──►
   └─INTFID-interface_id─────┘  │          ┌───────────────┐    │  └─MONSYSPLEX───┘
                                │          ▼               │    │
                                └─IPADDR────┤ ipaddr_spec ├┴────┘

   ┌─SECCLASS 255────────────┐
►──┼─────────────────────────┼──┬────────────────────────────────┬───────────────────►
   └─SECCLASS security_class─┘  └─SOURCEVIPAINTerface-vipa_name──┘

►──TRLEname trle_name─────────────────────────────────────────────────────────────────┤
```

ipaddr_spec:

```
├──┬─ipv6_address─────────────┬───────────────────────────────────────────────────────┤
   └─prefix/prefix_length─────┘
```

Parameters

intf_name
: The name of the interface. The maximum length is 16 characters.

  Restriction: Do not specify the value PUBLICADDRS or TEMPADDRS for the interface name. The values PUBLICADDRS and TEMPADDRS are keywords on the SRCIP statement. These values are not recognized if they are specified as an IPv6 interface name on an SRCIP entry.

DEFINE
: Specifies that this definition is to be added to the list of defined interfaces.

DELETE
: Specifies that this definition is to be deleted from the list of defined interfaces. The *intf_name* must be the name of an interface previously defined by an INTERFACE statement. INTERFACE DELETE deletes all home IP addresses for the interface.

ipaddr_spec
: For information about the IPv6 address restrictions, see "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143.

  The following value can be specified for ipaddr_spec:

- ipv6_addr (A fully qualified IPv6 address in colon-hexadecimal format)
- prefix/prefix_length. Here, the digits (in colon-hexadecimal format) before the / represent the prefix. The prefix length represents the length of the prefix in bits. If a prefix length is coded, it must be equal to 64. When a prefix is specified, TCP/IP forms the IPv6 address by appending an interface ID to the specified prefix. The selected interface ID is either the value specified by way of the INTFID keyword, or a random value that was generated at the time the interface was started.

ADDADDR *ipaddr_spec*
Allows the addition IP addresses to an existing INTERFACE definition (similar to updating the HOME list with a VARY TCPIP,,OBEYFILE command) without having to delete and redefine the INTERFACE. The *intf_name* coded with ADDADDR must be the name of an interface previously defined by an INTERFACE statement.

DELADDR *ipaddr_spec*
Allows you to delete IP addresses from an existing INTERFACE definition. The *intf_name* coded with DELADDR must be the name of an interface previously defined by an INTERFACE statement.

Guideline: If you specify a prefix for DELADDR, then the only IP addresses affected are those defined by way of the same prefix specified on IPADDR or ADDADDR.

DEPRADDR *ipaddr_spec*
Allows you to deprecate an IP address. This can assist with site renumbering. If you use DEPRADDR to deprecate an IP address, you can subsequently use ADDADDR to once again make that IP address preferred. The *intf_name* coded with DEPRADDR must be the name of an interface previously defined by an INTERFACE statement.

Guideline: If you specify a prefix for DEPRADDR, then the only IP addresses affected are those defined by way of the same prefix specified on IPADDR or ADDADDR.

MPCPTP6
Indicates that this interface operates as a MultiPath Channel connection for IPv6 traffic.

INTFID *interface_id*
An optional 64-bit interface identifier in colon-hexadecimal format. IPv6 shorthand is not allowed when specifying the interface ID. If specified, this interface ID is used to form the link-local address for the interface, and is also appended to any manually-configured prefixes for the interface, to form complete IPv6 addresses on the interface.

If INTFID is not coded, TCP/IP generates a random value to be used to form the link-local address. This random value is also used to complete the formation of other IPv6 addresses on the interface, if you choose to configure only the Prefix portion of the addresses (by way of IPADDR or ADDADDR).

For information about INTFID parameter restrictions, see “INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement” on page 161.

MONSYSPLEX | NOMONSYSPLEX
Specifies whether or not sysplex autonomics should monitor the interface's status.

NOMONSYSPLEX
Specifies that sysplex autonomics should not monitor the interface's status. This is the default value.

MONSYSPLEX
Specifies that sysplex autonomics should monitor the interface's status.

Restriction: The MONSYSPLEX attribute is not in effect unless the MONINTERFACE keyword is specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement. The presence of dynamic routes over this interface is monitored if the DYNROUTE keyword is also specified on the GLOBALCONFIG SYSPLEXMONITOR profile statement.

`SECCLASS` *`security_class`*
Use this parameter to associate a security class for IP filtering with this interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC6RULE statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: The TCP/IP stack ignores this value if the IPSECURITY parameter is not specified on the IPCONFIG6 statement.

`SOURCEVIPAINTERFACE` *`vipa_name`*
The SOURCEVIPAINTERFACE parameter is optional. It specifies which previously defined static VIPA interface is to be used for SOURCEVIPA (when IPCONFIG6 SOURCEVIPA is specified).

Tip: The use of the SOURCEVIPAINTERFACE parameter can be overridden. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

The *vipa_name* value is the interface name for a VIRTUAL6 interface. If the VIPA has multiple IP addresses, then the source VIPA address for outbound packets is selected from among these addresses according to the default source address selection algorithm. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.

Requirement: The VIRTUAL6 interface must be defined prior to specifying this INTERFACE statement to the TCP/IP stack. It must either already be defined or, the INTERFACE statement that defines it must precede this INTERFACE statement in the profile data set.

`TRLENAME` *`trle_name`*
The *trle_name* value must be the TRLE name of an HPDT connection. The TRLE is defined in a VTAM TRL major node and must be active before the interface can be started. For details about defining a TRLE, see z/OS Communications Server: SNA Resource Definition Reference.

The maximum length of the *trle_name* value is eight characters.

TRLE Name Specification for IP samehost, Enterprise Extender, and sysplex connections:

Specifying the reserved TRLE name IUTSAMEH allows for IPv6 communications over a samehost MPC point-to-point connection between the local TCP/IP stack and one or more TCP/IP stacks running on the same z/OS image. No physical device is needed to provide this connection between the stacks; VTAM provides a simulated communications link. VTAM dynamically defines the IUTSAMEH TRLE.

When running Enterprise Extender over an IPv6 network, you must define and start an MPCPTP6 interface, specifying the IUTSAMEH TRLE. This is not required if you specify IPCONFIG6 DYNAMICXCF.

For XCF connections, the *trle_name* must be the CP name of the target VTAM on the other side of the XCF connection, and the VTAM ISTLSXCF major node must be active in both nodes to start the device.

IPADDR *ipaddr_spec*
The IPADDR parameter is optional, and is used to configure the interface's IPv6 addresses other than the link-local address (which is generated internally by TCP/IP).

Rule: Stateless Address Autoconfiguration does not apply to MPCPTP6 interfaces, so you must manually configure any addresses (other than link-local) that are to be assigned to the MPCPTP6 interface.

Steps for modifying

See “Summary of INTERFACE statements” on page 141 for modification information.

Examples

```
INTERFACE MPCPTPV6A DEFINE MPCPTP6 TRLENAME ESCONCT1
     INTFID 0:0:0:1 IPADDR 12AB:0:0:0::/64
```

Usage notes

Requirement: IUTSAMEH definition is required if you plan to use the Enterprise Extender function over an IPv6 network, and the TCP/IP stack you are configuring is used for access to the IP network by VTAM on this host.

Restriction: A mix of static and dynamic IPv4 and IPv6 definitions for a device is not allowed. For example, if a static IUTSAMEH IPv4 device and link is defined, an IPv6 dynamic definition for IUTSAMEH is not created. If a static IUTSAMEH IPv6 interface is defined, an IPv4 dynamic definition for IUTSAMEH is not created. The same logic also applies for XCF links; a mix of static and dynamic IPv4 and IPv6 definitions is not allowed for an XCF link.

Rule: In order to configure a single physical device for both IPv4 and IPv6 traffic, you must use DEVICE/LINK/HOME for the IPv4 definition and INTERFACE for the IPv6 definition, such that the TRLENAME value on the INTERFACE statement matches the device_name on the DEVICE statement.

Related topics

- “BEGINROUTES statement” on page 28
- “DEVICE and LINK — MPCPTP devices statement” on page 92
- DYNAMICXCF in “IPCONFIG6 statement” on page 206
- “START statement” on page 292
- “STOP statement” on page 293

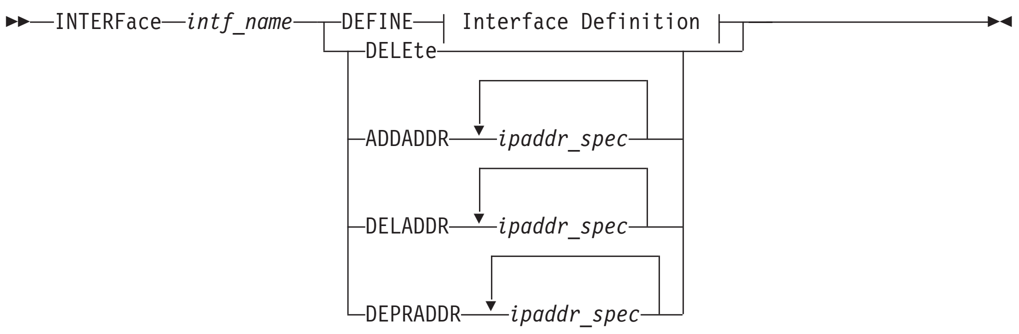
INTERFACE — VIRTUAL6 interfaces statement

Use the INTERFACE statement to specify a static virtual interface.

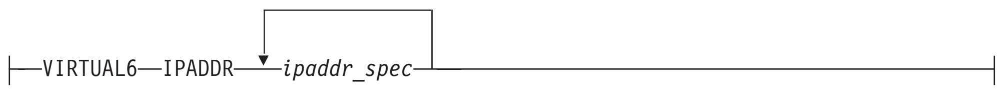
You can define multiple virtual IPv6 addresses on one TCP/IP image either by specifying multiple addresses on one VIRTUAL6 INTERFACE statement or by specifying multiple VIRTUAL6 INTERFACE statements.

Rule: Specify the required parameters in the order shown here. The optional parameters can be specified in any order.

Syntax



Interface Definition:



Parameters

intf_name
The name of the interface. The maximum length is 16 characters.

Restriction: Do not specify the value PUBLICADDRS or TEMPADDRS for the interface name. The values PUBLICADDRS and TEMPADDRS are keywords on the SRCIP statement. These values are not recognized if they are specified as an IPv6 interface name on an SRCIP entry.

DEFINE
Specifies that this definition is to be added to the list of defined interfaces.

DELETE
Specifies that this definition is to be deleted from the list of defined interfaces. The *intf_name* must be the name of an interface previously defined by an INTERFACE statement. INTERFACE DELETE deletes all home IP addresses for the interface.

ADDADDR *ipaddr_spec*
Allows the addition of IP addresses to an existing INTERFACE definition (similar to updating the HOME list with a VARY TCPIP,,OBEYFILE command), without having to delete and redefine the INTERFACE. The *intf_name* coded with ADDADDR must be the name of an interface previously defined by an INTERFACE statement.

If ADDADDR is specified, then *ipaddr_spec* can be one or more full IPv6 addresses.

ipaddr_spec
: For information about the IPv6 address restrictions, see "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143.

  The following value can be specified for ipaddr_spec:

  ipv6_address
  : IPv6 address in colon-hexadecimal format.

DELADDR *ipaddr_spec*
: Allows you to delete IP addresses from an existing INTERFACE definition. The *intf_name* coded with DELADDR must be the name of an interface previously defined by an INTERFACE statement.

  If DELADDR is specified, then *ipaddr_spec* can be one or more full IPv6 addresses.

  Restriction: You cannot code DELADDR and delete the last IP address for a VIRTUAL6 interface. You must delete the interface itself by specifying an INTERFACE statement for the interface name, with the DELETE parameter.

DEPRADDR *ipaddr_spec*
: The DEPRADDR keyword allows you to deprecate an IP address. This can assist with site renumbering. If you use DEPRADDR to deprecate an IP address, you can subsequently use ADDADDR again to make that IP address preferred. For DEPRADDR, the interface_name must be the name of an interface previously defined by an INTERFACE statement.

VIRTUAL6
: Indicates that the interface is not associated with real hardware and is used for fault tolerance support.

IPADDR *ipaddr_spec*
: This parameter is required and must be one or more full IPv6 addresses (no prefix is allowed).

Steps for modifying

See "Summary of INTERFACE statements" on page 141 for modification information.

Examples

```
INTERFACE VIPAV6 DEFINE
VIRTUAL6
IPADDR 12AB::1
12AB::2
```

Usage notes

- The TCP/IP stack does not maintain interface counters for VIRTUAL6 interfaces.
- A VIRTUAL6 interface name cannot be coded in the BEGINROUTES block.

Related topics

"IPCONFIG6 statement" on page 206

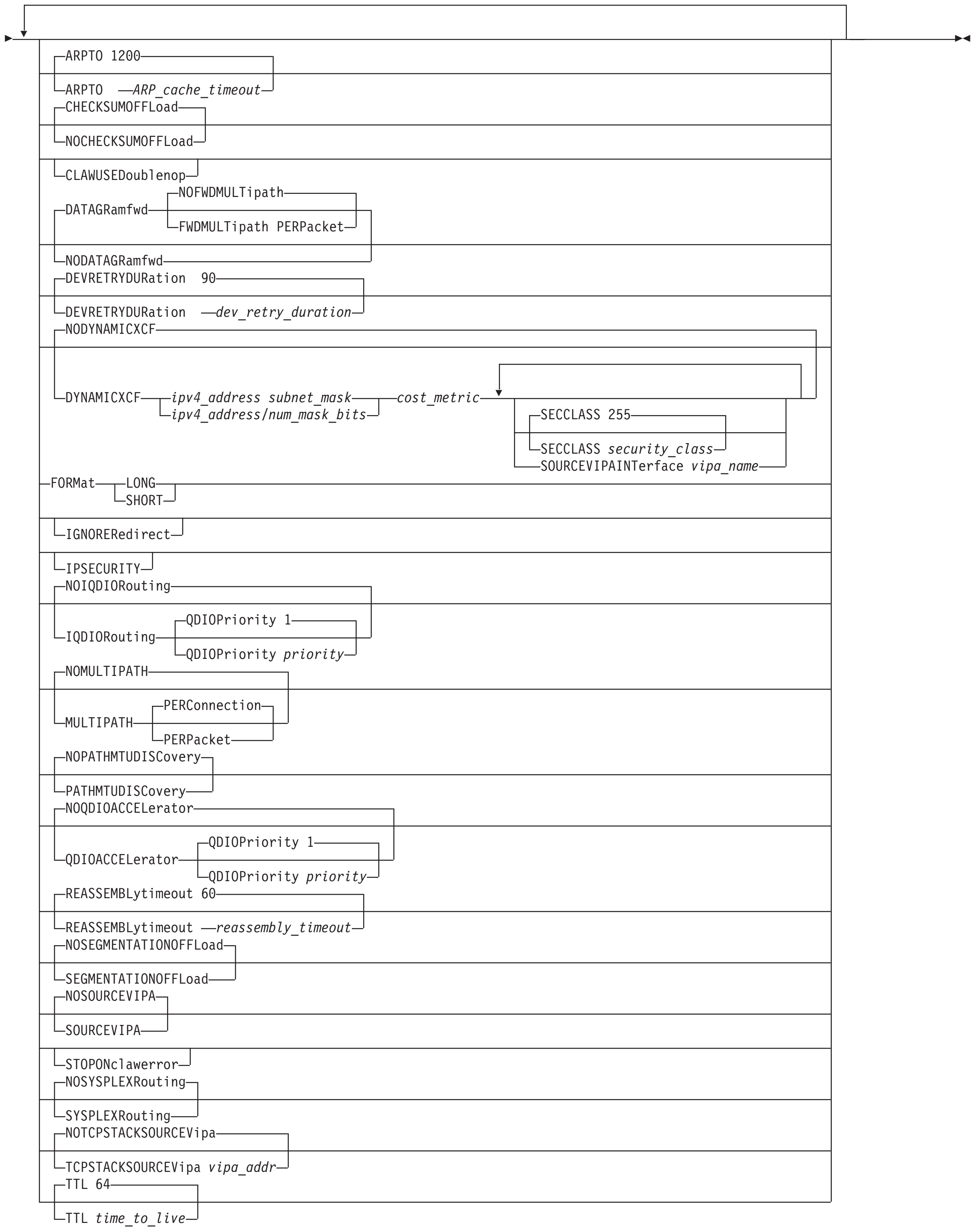
IPCONFIG statement

Use the IPCONFIG statement to update the IPv4 IP layer of TCP/IP.

Syntax

Tip: Specify the parameters for this statement in any order.

```
►►──IPCONFig──────────────────────────────────────────────────────────────────────►
```



Parameters

ARPTO *ARP_cache_timeout*
Use ARPTO to specify the number of seconds between creation or revalidation

and deletion of ARP table entries. The default is 1200 seconds. An LCS ARP table entry is revalidated when another ARP packet is received from the same host specifying the same hardware address. The minimum value is 60, and the maximum value is 86400.

This parameter serves the same purpose as the ARPAGE statement, but the value specified on ARPAGE is in minutes while the value specified on the ARPTO parameter is in seconds.

Because ARP cache entries for MPCIPA and MPCOSA interfaces are not managed by the TCP/IP stack, they are not affected by the ARPTO statement. For more information about devices that support ARP Offload, see z/OS Communications Server: IP Configuration Guide.

CHECKSUMOFFLOAD | NOCHECKSUMOFFLOAD

Specifies whether the stack should offload inbound and outbound checksum processing (IP header, TCP, and UDP checksums) for IPv4 packets to OSA-Express features. The checksum offload support transfers the overhead of most checksum processing to QDIO-attached OSA-Express devices that support this function. Offloading checksum processing reduces CPU use and increases throughput. This parameter is ignored for OSA-Express features that do not support checksum offload.

See “Steps for modifying” on page 204 for information about changing this parameter while the TCP/IP stack is active. See Checksum offload in z/OS Communications Server: IP Configuration Guide for more information about the checksum offload support and for specific information about which packets can have checksum processing performed by the OSA-Express.

NOCHECKSUMOFFLOAD

Checksum processing is performed by the TCP/IP stack.

CHECKSUMOFFLOAD

Checksum processing is offloaded to the OSA-Express feature. This value is the default value.

CLAWUSEDOUBLENOP

Forces channel programs for CLAW devices to have two NOP CCWs to end the channel programs. This is required for some vendor devices, and applies to only first-level MVS systems. The CLAWUSEDOUBLENOP parameter is confirmed by the message:

```
EZZ0337I CLAWUSEDOUBLENOP IS SET
```

DATAGRAMFWD | NODATAGRAMFWD

NODATAGRAMFWD

Disables the forwarding of IP packets that are received by, but not addressed to, the stack. This statement can be used for security or to ensure correct usage of limited resources. The NODATAGRAMFWD parameter is confirmed by the message:

```
EZZ0334I IP FORWARDING IS DISABLED
```

DATAGRAMFWD

Enables the forwarding of IP packets that are received by, but not addressed to, the stack. This is the default value.

Tip: The FWDMULTIPATH and NOFWDMULTIPATH keywords used with DATAGRAMFWD are independent of the MULTIPATH keyword on the IPCONFIG statement.

NOFWDMULTIPATH

When forwarding is in effect and there are multiple equal-cost routes to the destination and the NOFWDMULTIPATH parameter is specified, TCP/IP uses the first active route found for forwarding each IP packet. This is the default value. The DATAGRAMFWD NOFWDMULTIPATH parameter is confirmed by the message:

```
EZZ0641I IP FORWARDING NOFWDMULTIPATH SUPPORT IS ENABLED
```

FWDMULTIPATH PERPACKET
When forwarding is in effect and there are multiple equal-cost routes to the destination and the FWDMULTIPATH PERPACKET parameter is specified, TCP/IP selects a route for forwarding each IP packet on an approximate round-robin basis from the multiple equal-cost routes. The selected route is used for routing that IP packet. Connection or connectionless-oriented IP packets using the same destination address do not always use the same route, but they do use all possible active routes to that destination host. All IP packets for a given association with a destination host are spread across the multiple equal-cost routes. The DATAGRAMFWD FWDMULTIPATH PERPACKET parameter is confirmed by the message:

```
EZZ0641I IP FORWARDING FWDMULTIPATH PERPACKET SUPPORT IS ENABLED
```

Guideline: If the TCP/IP stack is also configured to be a sysplex distributor (see “VIPADYNAMIC statement summary” on page 306 for more information), datagrams destined to a sysplex-distributed dynamic VIPA are forwarded to stacks, whether or not forwarding is enabled.

DEVRETRYDURATION *dev_retry_duration*
Specifies the duration (in seconds) of the retry period for a failed device or interface. TCP/IP performs reactivation attempts at 30 second intervals during this retry period. The default for DEVRETRYDURATION is 90 seconds. A specification of 0 generates an infinite recovery period, which means reactivation attempts are performed until the device or interface is either successfully reactivated or manually stopped (by way of the VARY TCPIP,,STOP command, or the VARY TCPIP,,OBEYFILE command with a data set containing the STOP profile statement). The maximum specifiable value is 4294967295.

Guideline: The default 90–seconds retry duration is sufficient for transparent recovery following many types of device or channel errors. However, certain ESCON-attached routers cannot complete a microcode load in 90 seconds and installations might want to increase the DEVRETRYDURATION to automatically recover the device following these longer outages. On the other hand, installations running extensive automation built upon SNMP status and alerts can choose to code a small (nonzero) value in DEVRETRYDURATION, such that device recovery is deferred to external automation software, rather than a function of TCP/IP. For IPv4 interfaces that are defined with DEVICE and LINK statements, see also the AUTORESTART parameter in “Overview of DEVICE and LINK statements” on page 47. For IPv6 interfaces and IPv4 interfaces that are defined with the INTERFACE statement, the autorestart function is always active.

DYNAMICXCF | NODYNAMICXCF
Indicates XCF support status.

NODYNAMICXCF
Indicates XCF dynamic support is not enabled. The NODYNAMICXCF parameter is confirmed by the message:

```
EZZ0624I DYNAMIC XCF DEFINITIONS ARE DISABLED
```

NODYNAMICXCF is the default value.

DYNAMICXCF

Indicates that dynamic XCF support is enabled for IPv4.

When DYNAMICXCF is coded in the profile, the purpose is to generate dynamic XCF interfaces, if possible. When TCP/IP is active, but ISTLSXCF is not active, dynamic creation is deferred. Later, when a TCP/IP command such as VARY TCPIP,,OBEYFILE or VARY TCPIP,,START is executed, triggering profile processing, the stack again checks to see if ISTLSXCF is active. If ISTLSXCF is active at that time, then the dynamic XCF interfaces are generated.

Dynamic XCF definitions are not generated if there is a DEVICE and LINK definition with the same device or link name that dynamic XCF would generate, or if there is an INTERFACE definition with the same interface name that dynamic XCF would generate.

Activation of dynamic XCF interfaces is delayed if VTAM is not up or if OMPROUTE is not up and DELAYJOIN is coded on the GLOBALCONFIG SYSPLEXMONITOR statement. For more information about connectivity problems in a sysplex, see z/OS Communications Server: IP Configuration Guide.

When using dynamic XCF for Sysplex configuration, make sure that XCFINIT=YES or XCFINIT=DEFINE is coded in the VTAM start options, or if XCFINIT=NO was specified, ensure that a VARY ACTIVATE command is issued for the ISTLSXCF major node. This ensures that XCF connections between TCP stacks on different VTAM nodes in the sysplex can be established. See z/OS Communications Server: SNA Resource Definition Reference for directions to code the XCFINIT VTAM start option. The DISPLAY NET,VTAMOPTS command can be used to determine the XCFINIT setting.

cost_metric

Specifies the interface-level metric for the cost of use for the DYNAMICXCF interface. The *cost_metric* value is an integer in the range 0 - 14. If using OMPROUTE, the *cost_metric* value is overridden with a corresponding OMPROUTE interface parameter value that can be coded or set to the default value (Cost0= on OSPF_INTERFACE or In_Metric= on RIP_INTERFACE).

ipv4_address

The IP address to be used as the home address for all dynamically generated XCF, Same Host, and HiperSockets interfaces. A multicast address is not accepted in this case.

subnet_mask

Specifies the interface-level subnet mask for the DYNAMICXCF interface. If using OMPROUTE, the *subnet_mask* value is overridden with a corresponding OMPROUTE interface parameter value that can be coded or set to the default value.

/num_mask_bits

It is an integer value in the range 1 - 32 that represents the number of leftmost significant bits for the address mask.

SECCLASS *security_class*

Use this parameter to associate a security class for IP filtering with each dynamic XCF interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

This value is used only when IPSECURITY is specified on the IPCONFIG statement.

SOURCEVIPAINTERFACE *vipa_name*

The SOURCEVIPAINTERFACE parameter is optional. This parameter specifies which static VIPA interface is to be used as the source IP address when IPCONFIG SOURCEVIPA is specified and outbound packets are sent over the dynamically generated XCF, Same Host, or HiperSockets interfaces. The *vipa_name* value is the interface name for a VIRTUAL interface. The maximum length is 16 characters.

The use of the SOURCEVIPAINTERFACE parameter can be overridden. See Source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

Requirement: The VTAM ISTLSXCF major node must be active for XCF dynamics to work, except for the following scenarios:

- Multiple TCP/IP stacks on the same MVS image; a dynamic samehost definition is generated whether ISTLSXCF is active or not.
- HiperSockets is configured and enabled across multiple z/OS systems that are in the same sysplex and the same CEC; a dynamic IUTIQDIO link is created whether ISTLSXCF is active or not.

For information about activating the ISTLSXCF major node, see z/OS Communications Server: SNA Resource Definition Reference.

Restriction: A mix of static and dynamic IPv4 and IPv6 definitions for a device are not allowed. For example, if a static IUTSAMEH IPv4 interface is defined, then the IPv6 dynamic definition for IUTSAMEH is not created. If a static IUTSAMEH IPv6 interface is defined, then the IPv4 dynamic definition for IUTSAMEH is not created. The same logic is also applied for XCF interfaces; a mix of static and dynamic IPv4 and IPv6 definitions is not allowed for an XCF interface.

Guidelines:

1. Dynamic XCF can be enabled even in a single system sysplex. HiperSockets can be used between LPARs on the same central processor complex (CPC) even when MVS images in those LPARs are not defined to be part of the same sysplex. HiperSockets can also be used between LPARs even when some of those other LPARs are running Linux, as long as all of the stacks

connecting to HiperSockets and needing to exchange IP packets with each other define IP addresses that are all in the same subnet (as defined by the dynamic XCF IP address and subnet mask in the IPCONFIG DYNAMICXCF profile statement).

2. If the DYNAMICXCF parameter is added (using a VARY TCPIP,,OBEYFILE command data set) after the TCP/IP stack and OMPROUTE are active, the DYNAMICXCF link should be configured to OMPROUTE prior to issuing the VARY TCPIP,,OBEYFILE command. If you do not do this, the network mask is used as the subnet mask for the interface.

For more details about the use of DYNAMICXCF, see the DYNAMICXCF information in z/OS Communications Server: IP Configuration Guide. The DYNAMICXCF parameter is confirmed by the message:

```
EZZ0624I DYNAMIC XCF DEFINITIONS ARE ENABLED
```

FORMAT

The FORMAT keyword is optional, and there is no default.

The FORMAT keyword is meaningful only for stacks that are not enabled for IPv6. It controls the format of the command output. If FORMAT SHORT is specified and the stack is enabled for IPv6, then an error message is displayed. If the stack is not enabled for IPv6 and the user specified LONG format, the command output is displayed as if it could contain IPv6 addresses. If the stack is not enabled for IPv6 and the user specified SHORT format or did not specify the FORMAT keyword, then the command output is displayed as if it could contain only IPv4 addresses and not the longer IPv6 addresses.

If the stack is enabled for IPv6, then specifying the FORMAT keyword does not make any difference to the command format

IGNOREREDIRECT

Causes TCP/IP to ignore ICMP Redirect packets. The IGNOREREDIRECT parameter is confirmed by the message:

```
EZZ0335I ICMP WILL IGNORE REDIRECTS
```

If you are using OMPROUTE and you have IPv4 interfaces configured to OMPROUTE and this option is not specified, IGNOREREDIRECT is enabled automatically.

If you are using intrusion detection services (IDS) policy to detect and discard ICMP Redirects and this option is not specified, ICMP Redirects are discarded anyway while the policy is active.

If this option is not specified, and an ICMP redirect is received for a destination for which there is a HOST route in the routing table, then the original route is deleted and replaced by the redirect. This applies to all routes, including static routes.

IPSECURITY

Activates IPv4 IP filtering and IPv4 IPSec tunnel support.

Requirements:

- Use this parameter so that the stack can function with the Communications Server IKE daemon, and for the stack to receive IPv4 IPSec policy information such as IP filter rules from the policy agent.
- Use this parameter so that the stack can receive defensive filters from the Defense Manager daemon (DMD).

The IPSECURITY parameter is confirmed by the message:

```
EZZ0753I IPV4 SECURITY SUPPORT IS ENABLED
```

Restriction: IPSec functions can be activated only at initial activation of TCP/IP.

IQDIOROUTING | NOIQDIOROUTING

NOIQDIOROUTING

Specifies that inbound packets that are to be forwarded by this TCP/IP stack should not be routed directly between a HiperSockets device and an OSA-Express device in QDIO mode. These packets are processed and routed by this TCP/IP stack.

NOIQDIOROUTING is the default value. If NOIQDIOROUTING is explicitly specified, then the stack confirms that direct routing is disabled with the following message:

```
EZZ0688I IQDIO ROUTING IS DISABLED
```

IQDIOROUTING

Specifies that inbound packets that are to be forwarded by this TCP/IP stack are eligible to be routed directly between a HiperSockets device and an OSA-Express device in QDIO mode without needing to be sent to this TCP/IP stack for forwarding. This type of routing over a HiperSockets device (iQDIO) is called HiperSockets Accelerator. If specified, HiperSockets Accelerator routes are created dynamically as this TCP/IP stack learns of destination IP addresses that can be routed to or from HiperSockets links without needing to be forwarded to this TCP/IP stack. HiperSockets Accelerator support cannot be enabled if the IPSECURITY parameter or the NODATAGRAMFWD parameter is specified. Use of the IQDIOROUTING parameter is confirmed by the following message:

```
EZZ0688I IQDIO ROUTING IS ENABLED
```

If HiperSockets Accelerator support cannot be enabled, message EZZ0689I is issued with the reason. This message is also issued if IQDIOROUTING is specified in the data set that is used with the VARY TCPIP,,OBEYFILE command, if TCP/IP was activated with NOIQDIOROUTING and NOQDIOACCELERATOR on the initial profile.

Rule: This parameter is ignored if QDIOACCELERATOR is specified.

Restrictions:

- HiperSockets Accelerator support cannot be enabled during VARY TCPIP,,OBEYFILE command processing unless either IQDIOROUTING or QDIOACCELERATOR was specified on the IPCONFIG statement in the initial profile.
- HiperSockets Accelerator does not accelerate packets either from or to interfaces configured with optimized latency mode. For more information about optimized latency mode, see Optimized latency mode in z/OS Communications Server: IP Configuration Guide.
- You cannot enable HiperSockets accelerator support if you specify the NODATAGRAMFWD parameter.
- You cannot enable HiperSockets accelerator support if you specify the IPSECURITY parameter.

QDIOPRIORITY priority

If traffic is being routed by way of HiperSockets Accelerator, the data is sent using the priority level specified by priority. *priority* values are in the range 1 - 4. The default is to send data using priority level 1. See the OSA-Express documentation in z/OS Communications Server: SNA Network Implementation Guide.

QDIOACCELERATOR | NOQDIOACCELERATOR

NOQDIOACCELERATOR

Specifies that inbound packets that are to be forwarded by this TCP/IP stack should not be routed directly between any of the following combinations of interface types:

- A HiperSockets interface and an OSA-Express QDIO interface
- Two OSA-Express QDIO interfaces
- Two HiperSockets interfaces

These packets are processed and routed by this TCP/IP stack.

NOQDIOACCELERATOR is the default value. If NOQDIOACCELERATOR is explicitly specified, the stack confirms this type of routing with the message:

```
EZZ0817I QDIO ACCELERATOR IS DISABLED
```

QDIOACCELERATOR

Specifies that inbound packets that are to be forwarded by this TCP/IP stack are eligible to be routed directly between any of the following combinations of interface types:

- A HiperSockets interface and an OSA-Express QDIO interface
- Two OSA-Express QDIO interfaces
- Two HiperSockets interfaces

These packets do not need to be sent to this TCP/IP stack for forwarding. This also applies to packets that would be forwarded by the Sysplex Distributor. This type of routing is called QDIO Accelerator. See the information about QDIO Accelerator in z/OS Communications Server: IP Configuration Guide for more details on this function.

Use of the QDIOACCELERATOR parameter is confirmed by one of the following messages:

```
EZZ0817I QDIO ACCELERATOR IS ENABLED
EZZ0819I QDIO ACCELERATOR IS ENABLED FOR SYSPLEX DISTRIBUTOR ONLY
EZD2020A QDIO ACCELERATOR IS ENABLED ONLY FOR SYSPLEX DISTRIBUTOR BECAUSE OF TCPIP PROFILE FILTER RULES
EZD2021A QDIO ACCELERATOR IS ENABLED ONLY FOR SYSPLEX DISTRIBUTOR BECAUSE OF POLICY FILTER RULES
EZD2022A QDIO ACCELERATOR IS ENABLED ONLY FOR SYSPLEX DISTRIBUTOR BECAUSE OF DEFENSIVE FILTER RULES
EZD2023I QDIO ACCELERATOR IS ENABLED WITH CURRENTLY INSTALLED IP FILTER RULES
```

You receive the following message with the appropriate reason if QDIO Accelerator support cannot be enabled. You also receive this message if QDIOACCELERATOR is specified in the data set used with the VARY TCPIP,,OBEYFILE command, if TCP/IP was activated with NOIQDIOROUTING and NOQDIOACCELERATOR on the initial profile.

```
EZZ0818I CANNOT ENABLE QDIO ACCELERATOR - reason
```

Rule: IQDIOROUTING is ignored if QDIOACCELERATOR is specified.

Restrictions:

- If you specify the NODATAGRAMFWD parameter, then QDIO Accelerator applies only to packets that are forwarded by the Sysplex Distributor.
- If you specify the IPSECURITY parameter, TCP/IP monitors your IP filter rules and defensive filter rules that apply to routed traffic.

Depending on your filter configuration, QDIO Accelerator might be restricted to only packets that are forwarded by the Sysplex Distributor. For more information about QDIO Accelerator and IPSECURITY, see Search orders used in the native MVS environment in z/OS Communications Server: IP Configuration Guide.

- QDIO Accelerator support cannot be enabled during VARY TCPIP,,OBEYFILE command processing unless either QDIOACCELERATOR or IQDIOROUTING was specified on the IPCONFIG statement in the initial profile.
- QDIO Accelerator does not accelerate packets either from or to interfaces configured with optimized latency mode. For more information about optimized latency mode, see Optimized latency mode in z/OS Communications Server: IP Configuration Guide.

QDIOPRIORITY *priority*
Specifies that traffic routed by QDIO Accelerator to an OSA-Express QDIO interface be sent using the priority level specified by the *priority* value. The priority level can be in range 1 - 4. The default is to send data using priority level 1. See the OSA-Express information in the z/OS Communications Server: SNA Network Implementation Guide.

MULTIPATH | NOMULTIPATH

NOMULTIPATH
Disables the multipath routing selection algorithm for outbound IP traffic. If there are multiple equal-cost routes to a destination and NOMULTIPATH is specified, TCP/IP uses the first active route found to send each IP packet. The NOMULTIPATH parameter is confirmed by the message:

```
EZZ0615I MULTIPATH SUPPORT IS DISABLED
```

This is the default value.

Rule: The NOMULTIPATH parameter applies to outbound IP traffic that is routed by using the main route table. This parameter applies also to outbound IP traffic that is routed by using a policy-based route table if the Multipath UseGlobal parameter is specified on the RouteTable statement that defines the policy-based route table. See “RouteTable statement” on page 1152 for more information.

MULTIPATH
Enables the multipath routing selection algorithm for outbound IP traffic. In general, multipath routing provides the routing distribution necessary to balance the network utilization of outbound packets by load splitting. Multipath routing requires multiple equal-cost routes that are either defined statically or added dynamically by routing protocols (except for RIP, which does not provide multipath routing). If MULTIPATH is specified without any subparameters, the default is PERCONNECTION. The MULTIPATH parameter has no effect if there are no multipath routes in the TCP/IP configuration.

Guideline: In some cases, it might appear data is not being equally distributed among each of the equal-cost interfaces. This depends upon the characteristics of the application that is sending or receiving data. For example, when `osnmp walk` is issued, the application initially sends data using a source IP address of INADDR_ANY. Subsequently, when the application receives a response, all future sends use the source IP address of the interface where data was just received. The result is that all data is sent out on a single interface, independent of any multipath setting.

Rules:

- The MULTIPATH parameter and its subparameters apply to outbound IP traffic that is routed by using the main route table. This parameter and its subparameters apply also to outbound IP traffic that is routed by using a policy-based route table if the Multipath UseGlobal parameter is specified on the RouteTable statement that defines the policy-based route table. See "RouteTable statement" on page 1152 for more information.
- The multipath routing selection algorithm is applied and can be specified separately for each route table. Specify the algorithm for the main route table by using the MULTIPATH parameter on the IPCONFIG statement. Specify the algorithm for policy-based route tables in the policy definition for each table. See "RouteTable statement" on page 1152 for more information.

Note: The IPCONFIG MULTIPATH|NOMULTIPATH configuration option affects Enterprise Extender (EE) traffic when MULTPATH=TCPVALUE is coded. For information about multipath for EE see z/OS Communications Server: SNA Network Implementation Guide. For information about the MULTPATH start option, see z/OS Communications Server: SNA Resource Definition Reference.

PERCONNECTION

After a round-robin route is selected, connection or connectionless oriented IP packets using the same association always use the same route, as long as that route is active. The MULTIPATH PERCONNECTION parameter is confirmed by the message:

```
EZZ0632I MULTIPATH PERCONNECTION SUPPORT IS ENABLED
```

For more information about EE load balancing and standard logic for a UDP application, see z/OS Communications Server: SNA Network Implementation Guide.

PERPACKET

Connection or connectionless oriented IP packets using the same source and destination address pair do not always use the same route, but do use all possible active routes to that destination host. The MULTIPATH PERPACKET parameter is confirmed by the message:

```
EZZ0632I MULTIPATH PERPACKET SUPPORT IS ENABLED
```

Restrictions:

- Use this option only as an attempt to improve aggregate throughput of IP traffic over multipath routes and for routes for which potentially high CPU consumption in reassembly of out-of-order packets at the receiving end is not an issue. Performance varies according to network configurations used.

- The MULTIPATH PERPACKET parameter cannot be specified if IP security is configured. If both are specified, the following messages are displayed, and multipath routing is disabled:

```
EZZ0763I CANNOT ENABLE IPV4 MULTIPATH PERPACKET SUPPORT WHEN
         IPV4 SECURITY IS ENABLED
EZZ0615I MULTIPATH SUPPORT IS DISABLED
```

- IP traffic on RSVP-based routes cannot use this option. Instead, the PERCONNECTION option is used for these routes.
- Fragmented and packed IP datagrams cannot use this option. These datagrams are being sent over one selected route to the intended destination.

PATHMTUDISCOVERY | NOPATHMTUDISCOVERY

NOPATHMTUDISCOVERY
Indicates that TCP/IP is not to provide path MTU (PMTU) discovery support. This is the default value. The NOPATHMTUDISCOVERY parameter is confirmed by the message:

```
EZZ0623I PATH MTU DISCOVERY SUPPORT IS DISABLED
```

PATHMTUDISCOVERY
Indicates that TCP/IP is to dynamically discover the PMTU, which is the smallest MTU of all the hops in the path. Use this parameter to prevent fragmentation of datagrams. The PATHMTUDISCOVERY parameter is confirmed by the message:

```
EZZ0623I PATH MTU DISCOVERY SUPPORT IS ENABLED
```

Requirement: PATHMTUDISCOVERY uses ICMP fragmentation-needed errors to detect the PMTU for a path. If you use PATHMTUDISCOVERY, you must permit ICMP errors to flow at all hosts along the path of a connection. PATHMTUDISCOVERY does not function if a firewall blocks ICMP errors.

For a policy-based route table, the IgnorePathMtuUpdate parameter on the Policy Agent RouteTable statement can be used to prevent the path MTU value from being updated for routes in the table. See the information about the IgnorePathMtuUpdate parameter in "RouteTable statement" on page 1152 for information about determining when you should prevent the path MTU value from being updated for a policy-based route table.

REASSEMBLYTIMEOUT *reassembly_timeout*
The amount of time (in seconds) allowed to receive all parts of a fragmented packet before the fragments received are discarded. The minimum value is 1, the maximum value is 240, and the default is 60.

SEGMENTATIONOFFLOAD | NOSEGMENTATIONOFFLOAD
Specifies whether the stack should offload TCP segmentation for IPv4 packets to OSA-Express features. The TCP segmentation offload support transfers the overhead of segmenting outbound data into individual TCP packets to QDIO-attached OSA-Express devices that support this function. Offloading segmentation of streaming-type workloads reduces CPU use and increases throughput. This parameter is ignored for OSA-Express features that do not support segmentation offload. This value overrides the SEGMENTATIONOFFLOAD or NOSEGMENTATIONOFFLOAD parameter specified on the GLOBALCONFIG statement.

See the steps for modifying topic for information about changing this parameter while the TCP/IP stack is active. See TCP segmentation offload in z/OS Communications Server: IP Configuration Guide for more information about TCP segmentation offload support.

NOSEGMENTATIONOFFLOAD
TCP segmentation is performed by the TCP/IP stack. This value is the default value.

SEGMENTATIONOFFLOAD
TCP segmentation is offloaded to the OSA-Express feature.

SOURCEVIPA | NOSOURCEVIPA

NOSOURCEVIPA
Specifies that TCP/IP is not requested to use the corresponding virtual IP address in the HOME list as the source IP address for outbound datagrams. The NOSOURCEVIPA parameter is confirmed by the message:

```
EZZ0351I SOURCEVIPA SUPPORT IS DISABLED.
```

NOSOURCEVIPA is the default value.

SOURCEVIPA
Requests that TCP/IP use the TCPSTACKSOURCEVIPA address (if specified) or the corresponding virtual IP address in the HOME list as the source IP address for outbound datagrams that do not have an explicit source address. If the outgoing interface was defined with the INTERFACE statement, TCP/IP uses the VIPA specified on the SOURCEVIPAINTERFACE parameter of the INTERFACE statement instead of the HOME list. You must specify the TCPSTACKSOURCEVIPA parameter, update the HOME statement, or use the SOURCEVIPAINTERFACE parameter of the INTERFACE statement for the SOURCEVIPA parameter to take effect. For more information about how the order of the HOME list impacts source VIPA selection, see "HOME statement" on page 136. This parameter has no effect on RIP packets used by RIP services (NCPROUTE or OMPROUTE) or OSPF packets used by OSPF services (OMPROUTE). The SOURCEVIPA parameter is confirmed by the following message:

```
EZZ0351I SOURCEVIPA SUPPORT IS ENABLED
```

Tip: You can override the SOURCEVIPA or TCPSTACKSOURCEVIPA values. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

STOPONCLAWERROR
Stops channel programs (HALTIO and HALTSIO) when a device error is detected. The STOPONCLAWERROR parameter is confirmed by the message:

```
EZZ0345I STOPONCLAWERROR IS ENABLED
```

SYSPLEXROUTING | NOSYSPLEXROUTING

NOSYSPLEXRouting
Specifies that this TCP/IP host is not part of an MVS sysplex domain. Use of the NOSYSPLEXROUTING parameter is confirmed by the message:

```
EZZ0350I SYSPLEX ROUTING SUPPORT IS DISABLED
```

NOSYSPLEXROUTING is the default value.

SYSPLEXRouting
Specifies that this TCP/IP host is part of an MVS sysplex domain. The SYSPLEXROUTING parameter is confirmed by the message:

```
EZZ0350I SYSPLEX ROUTING SUPPORT IS ENABLED
```

`TCPSTACKSOURCEVIPA | NOTCPSTACKSOURCEVIPA`

`NOTCPSTACKSOURCEVIPA`
Specifies that TCP/IP does not use a stack-level IP address as the source address for outbound TCP connections. The source IP address is governed by the IPCONFIG SOURCEVIPA setting.

`TCPSTACKSOURCEVIPA` *`vipa_addr`*
The IPv4 address (*vipa_addr*) is used as the source IP address for outbound TCP connections if SOURCEVIPA has been enabled. The *vipa_addr* value must be a static VIPA or an active dynamic VIPA (DVIPA).

If SOURCEVIPA has not been enabled, TCPSTACKSOURCEVIPA is ignored, and the following message is issued:

```
EZZ0706I TCPSTACKSOURCEVIPA IS IGNORED - SOURCEVIPA IS NOT ENABLED
```

Restriction: At the time of an outbound TCP request, the TCPSTACKSOURCEVIPA address must be a static VIPA or active dynamic VIPA, or it is not used for the source IP address.

Tips:

- After it is set, TCPSTACKSOURCEVIPA is not disabled until a profile explicitly adds NOTCPSTACKSOURCEVIPA to the IPCONFIG statement.
- If you specify the same distributed DVIPA interface for TCPSTACKSOURCEVIPA on multiple target stacks, you also should specify SYSPLEXPORTS on the VIPADISTRIBUTE statement. Otherwise connections might be disrupted because identical connections could be created from more than one stack.
- Carefully consider the following condition when determining the interface to use for TCPSTACKSOURCEVIPA. A dynamic VIPA that becomes inactive because it moves to another TCP/IP stack, or that is deleted because the application that caused its creation (in the case of a VIPARANGE created address) causes its deletion, is no longer a valid interface for TCPSTACKSOURCEVIPA.
- The use of TCPSTACKSOURCEVIPA can be overridden. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.
- TCPSTACKSOURCEVIPA is not used when an outbound TCP request is connecting to an IP address that is active in the Home list.

`TTL` *`time_to_live`*
Number of hops that packets originating from this host can travel before reaching the destination. If the destination is more hops away, the packet never reaches the destination. The minimum value is 1, the maximum value is 255, and the default is 64.

Steps for modifying

To modify most parameters for the IPCONFIG statement, you must respecify the statement with the new parameters. Additional actions are required to modify the following parameters:

CHECKSUMOFFLOAD | NOCHECKSUMOFFLOAD
: If the CHECKSUMOFFLOAD or NOCHECKSUMOFFLOAD parameter is changed with the VARY TCPIP,,OBEYFILE command, the new value does not affect any active OSA-Express QDIO interfaces. For this change to affect an active OSA-Express QDIO interface, the interface must be stopped and restarted.

DYNAMICXCF
: If dynamic XCF definitions have been enabled but a later VARY TCPIP,,OBEYFILE command contains NODYNAMICXCF, only future dynamic definitions and connectivity are affected. Existing definitions and connectivity are not affected.

  After support is enabled, none of the parameters specified on the IPCONFIG DYNAMICXCF statement can be changed with a VARY TCPIP,,OBEYFILE command. You must first stop the TCP/IP stack, apply changes, and then restart the TCP/IP stack.

IPSECURITY
: z/OS IPSec functions cannot be activated using VARY TCPIP,,OBEYFILE on an active TCP/IP stack. To activate z/OS IPSec, halt all traffic on the designated TCP/IP stack, stop the stack, modify the TCP profile to include IPCONFIG IPSECURITY, and restart the stack.

IQDIOROUTING
: If HiperSockets Accelerator is active then:

  - You can disable HiperSockets Accelerator by issuing the VARY TCPIP,,OBEYFILE command and specifying IPCONFIG NOIQDIOROUTING.
  - You can activate QDIO Accelerator by issuing the VARY TCPIP,,OBEYFILE command and specifying IPCONFIG NOIQDIOROUTING QDIOACCELERATOR.

  If HiperSockets Accelerator and QDIO Accelerator are not active and you want to enable HiperSockets Accelerator, enable HiperSockets Accelerator by issuing the VARY TCPIP,,OBEYFILE command and specifying IPCONFIG IQDIOROUTING (if either IQDIOROUTING or QDIOACCELERATOR was specified in the initial profile); otherwise, stop the stack, modify the profile to include IPCONFIG IQDIOROUTING and restart the stack.

NODATAGRAMFWD
: If HiperSockets Accelerator is enabled and IP forwarding is subsequently disabled by issuing a VARY TCPIP,,OBEYFILE with NODATAGRAMFWD specified, HiperSockets Accelerator is also disabled. If HiperSockets Accelerator is disabled, and IPCONFIG IQDIOROUTING is subsequently specified on a VARY TCPIP,,OBEYFILE command for an active TCP/IP stack where IP Forwarding is disabled, HiperSockets Accelerator remains disabled.

  If QDIO Accelerator is enabled and IP Forwarding is subsequently disabled using NODATAGRAMFWD in a VARY TCPIP,,OBEYFILE command data set, QDIO Accelerator remains enabled but only for Sysplex Distributor forwarding. If QDIO Accelerator is disabled and IPCONFIG QDIOACCELERATOR is subsequently specified on a VARY TCPIP,,OBEYFILE command for an active TCP/IP stack on which IP forwarding is disabled, QDIO Accelerator is enabled for Sysplex Distributor forwarding only.

QDIOACCELERATOR

If QDIO Accelerator is active:

- You can disable QDIO Accelerator by issuing the VARY TCPIP,,OBEYFILE command and specifying IPCONFIG NOQDIOACCELERATOR.
- You can activate HiperSockets Accelerator by issuing the VARY TCPIP,,OBEYFILE command and specifying IPCONFIG NOQDIOACCELERATOR IQDIOROUTING.

If QDIO Accelerator and HiperSockets Accelerator are not active and you want to enable QDIO Accelerator, enable QDIO Accelerator by issuing the VARY TCPIP,,OBEYFILE command and specifying IPCONFIG QDIOACCELERATOR (if either IQDIOROUTING or QDIOACCELERATOR was specified in the initial profile); otherwise, stop the stack, modify the profile to include IPCONFIG QDIOACCELERATOR and restart the stack.

MULTIPATH

If you modify the multipath routing type (PERCONNECTION to PERPACKET, or vice versa), the new parameter takes effect only for new connections created after the modification is done, and existing connections use whatever the value was when the connection was established. If you enable multipath routing when it was previously disabled, existing connections are not affected; multipath routing is applied only to new connections.

SEGMENTATIONOFFLOAD | NOSEGMENTATIONOFFLOAD

If the SEGMENTATIONOFFLOAD or NOSEGMENTATIONOFFLOAD parameter is changed with the VARY TCPIP,,OBEYFILE command, the new value does not affect any active OSA-Express QDIO interfaces. For this change to affect an active OSA-Express QDIO interface, the interface must be stopped and restarted.

Examples

```
IPCONFIG ARPTO 2400 CLAWUSED NODATAGR STOPON
DYNAMICXCF 9.9.9.9 255.255.255.0 15
```

This example shows an IPCONFIG statement that does the following tasks:

- Causes ARP table entries to be deleted 2400 seconds after creation or revalidation
- Forces channel programs for CTC devices to have two NOP CCWs to end the channel programs
- Disables IP forwarding
- Causes TCP/IP to halt on certain CLAW errors
- Enables dynamic XCF support and indicates that 9.9.9.9 is the IP address to be used for HOME statements for all dynamically generated XCF, Same Host, and HiperSockets links. These links have an interface-level subnet mask of 255.255.255.0 and a metric of 15.

Usage notes

- If the stack is enabled for IPv6 and the user specified LONG format, the command output is displayed in IPv6 format.
- The FORMAT keyword is meaningful only for stacks that are not enabled for IPv6. It controls the format of the command output. If FORMAT SHORT is specified and the stack is enabled for IPv6, then the following error message is displayed:

  ```
  EZZ0687I FORMAT SHORT IGNORED - IPV6 SUPPORT IS ENABLED
  ```
- If you do not include any configuration data in the OMPROUTE configuration file for the XCF links, OMPROUTE does not communicate a routing protocol (OSPF or RIP) over the interfaces. OMPROUTE includes (in the data sent to other routers) information relative to the XCF links as long as Send_Static_Routes=YES is configured for RIP Interfaces and AS_Boundary_Routing(Import_Static_Routes=YES) is configured for OSPF.

  Rule: If you want to communicate the OSPF or RIP protocol over a subset of the XCF links, you must configure the appropriate links in the OMPROUTE configuration file using the OSPF_Interface or RIP_Interface statements. Doing this enables OMPROUTE to communicate to other routers not only the information relative to the XCF links, but also information relative to resources on the other side of the host at the opposite end of the XCF links.

  To configure the appropriate links, you can explicitly configure each XCF link as either an OSPF or RIP interface (including those that might become active in the future). Alternatively, you can use the wildcard configuration capability of OMPROUTE to configure your XCF links.

  To use the wildcard configuration, use a wildcard address (for example, 9.67.100.*) on the OSPF_Interface or RIP_Interface statement instead of an explicit address. In this way, any interface address falling within that wildcard range (9.67.100.1, 9.67.100.2, and so on) is configured using the parameters specified on the wildcard definition statement.

  When adding links, XCF or otherwise, to both OMPROUTE and TCP/IP, it is necessary to add them to OMPROUTE before adding them to TCP/IP for proper routing protocol configuration.

Related topics

- "GLOBALCONFIG statement" on page 117
- "IPCONFIG6 statement"
- "SRCIP statement" on page 282

IPCONFIG6 statement

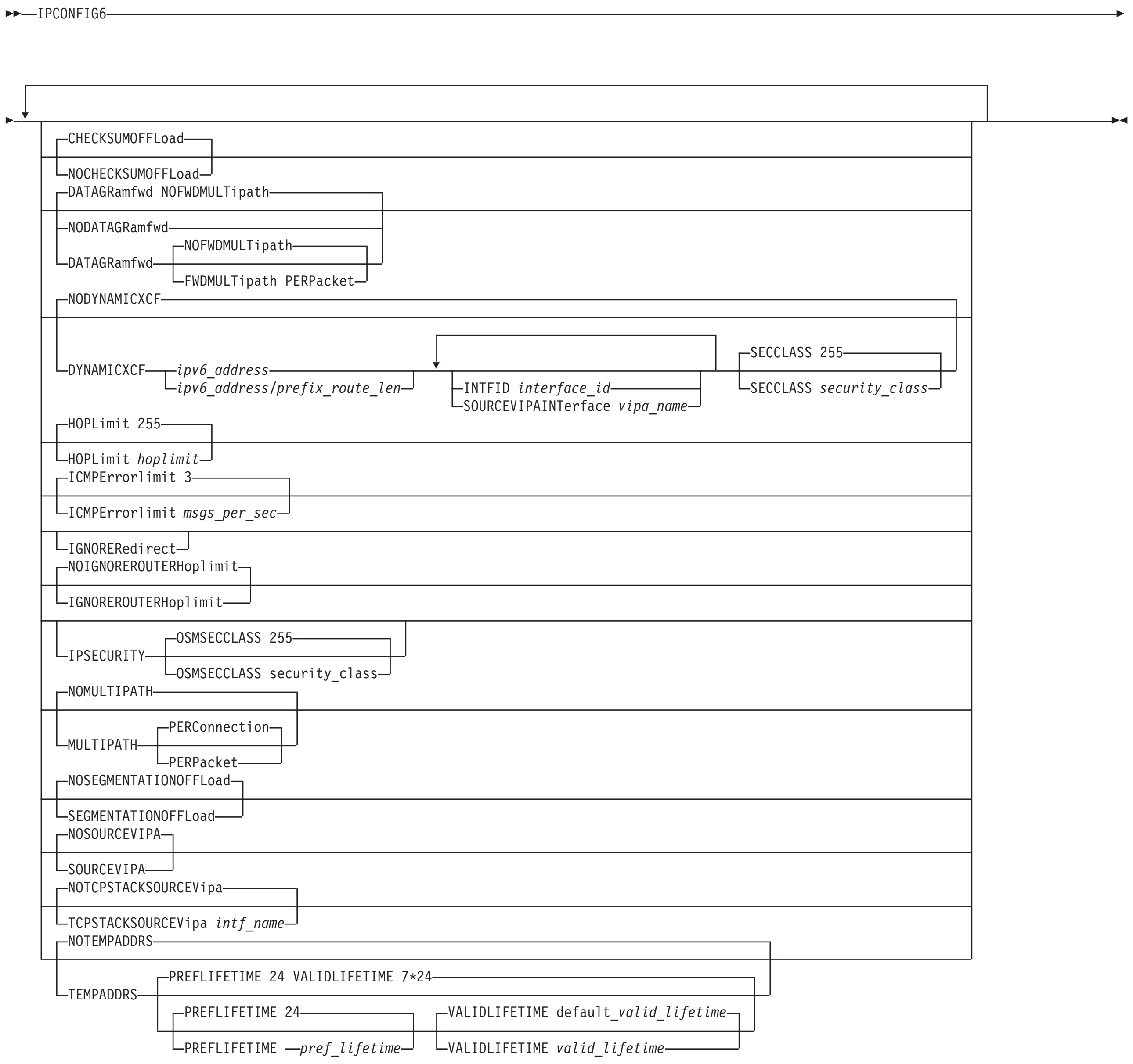
Use the IPCONFIG6 statement to update the IP layer of TCP/IP with information that pertains to IPv6.

If the stack is not configured for IPv6 and IPCONFIG6 is specified, the following error message is generated, and TCP/IP startup processing continues.

```
EZZ0695I IPCONFIG6 NOT VALID - IPV6 SUPPORT IS NOT ENABLED
```

Syntax

Tip: Specify the parameters for this statement in any order.



Parameters

CHECKSUMOFFLOAD | NOCHECKSUMOFFLOAD

Specifies whether the stack should offload inbound and outbound checksum processing (TCP and UDP checksums) for IPv6 packets to OSA-Express features. The checksum offload support transfers the overhead of most checksum processing to QDIO-attached OSA-Express devices that support this function. Offloading checksum processing reduces CPU use and increases throughput. This parameter is ignored for OSA-Express features that do not support IPv6 checksum offload.

See "Steps for modifying" on page 217 for information about changing this parameter while the TCP/IP stack is active. See Checksum offload in z/OS Communications Server: IP Configuration Guide for more information about the checksum offload support and for specific information about which packets can have checksum processing performed by the OSA-Express.

NOCHECKSUMOFFLOAD
Checksum processing is performed by the TCP/IP stack.

CHECKSUMOFFLOAD
Checksum processing is offloaded to the OSA-Express feature. This value is the default value.

DATAGRAMFWD | NODATAGRAMFWD

NODATAGRAMFWD
Disables the forwarding of IP packets that are received by, but not addressed to, the stack. This statement can be used for security or to ensure correct usage of limited resources. The NODATAGRAMFWD parameter is confirmed by the message:

```
EZZ0699I IPV6 FORWARDING IS DISABLED
```

If the TCP/IP stack is also configured to be a sysplex distributor (see "VIPADYNAMIC statement summary" on page 306 for more information), datagrams destined to a sysplex-distributed dynamic VIPA are forwarded to stacks, whether or not forwarding is enabled.

DATAGRAMFWD
Enables the forwarding of IP packets that are received by, but not addressed to, the stack. This is the default value.

NOFWDMULTIPATH
When forwarding is in effect and there are multiple equal-cost routes to the destination and the NOFWDMULTIPATH parameter is specified, TCP/IP uses the first active route found for forwarding each IP packet. This is the default value. The DATAGRAMFWD NOFWDMULTIPATH parameter is confirmed by the message:

```
EZZ0700I IPV6 FORWARDING NOFWDMULTIPATH SUPPORT IS ENABLED
```

FWDMULTIPATH PERPACKET
When forwarding is in effect and there are multiple equal-cost routes to the destination and the FWDMULTIPATH PERPACKET parameter is specified, TCP/IP selects a route for forwarding each IP packet on an approximate round-robin basis from the multiple equal-cost routes. Connection or connectionless-oriented IP packets using the same destination address do not always use the same route, but they do use all possible active routes to that destination host. All IP packets for a given association with a destination host are spread across the multiple equal-cost routes. The DATAGRAMFWD FWDMULTIPATH PERPACKET parameter is confirmed by the message:

```
EZZ0700I IPV6 FORWARDING FWDMULTIPATH PERPACKET SUPPORT IS ENABLED
```

DYNAMICXCF | NODYNAMICXCF

NODYNAMICXCF
Indicates XCF dynamic support is not enabled for IPv6 on this TCP/IP. The NODYNAMICXCF parameter for IPCONFIG6 is confirmed by the message:

```
EZZ0739I IPV6 DYNAMIC XCF DEFINITIONS ARE DISABLED
```

DYNAMICXCF
Indicates that dynamic XCF support is enabled for IPv6.

When DYNAMICXCF is coded in the profile, the purpose is to generate those dynamic XCF devices or interfaces, if possible. When TCP/IP is up, but ISTLSXCF is not active, dynamic creation is deferred. Later, when a TCP/IP command such as VARY TCPIP,,OBEYFILE or VARY TCPIP,,START is executed, triggering profile processing, the stack again checks to see if ISTLSXCF is active. If ISTLSXCF is active at that time, then the dynamic XCF devices and interfaces are generated.

Dynamic XCF definitions are not generated if there is a DEVICE or INTERFACE definition with the same device or interface name that dynamic XCF would generate.

Activation of dynamic XCF links is delayed if VTAM is not up or if OMPROUTE is not up and DELAYJOIN is coded on the GLOBALCONFIG SYSPLEXMONITOR statement. For more information about connectivity problems in a sysplex, see z/OS Communications Server: IP Configuration Guide.

When using dynamic XCF for sysplex configuration, make sure that XCFINIT=YES or XCFINIT=DEFINE is coded in the VTAM start options, or if XCFINIT=NO was specified, ensure that a VARY ACTIVATE command is issued for the ISTLSXCF major node. This ensures that XCF connections between TCP stacks on different VTAM nodes in the sysplex can be established. See z/OS Communications Server: SNA Resource Definition Reference for directions for coding the XCFINIT VTAM start option. The DISPLAY NET,VTAMOPTS command can be used to determine the XCFINIT setting.

The VTAM ISTLSXCF major node must be active for XCF dynamics to work, except for the following two scenarios:

- Multiple TCP/IP stacks on the same MVS image; a dynamic samehost definition is generated, whether ISTLSXCF is active or not.
- HiperSockets is configured and enabled across multiple z/OS systems that are in the same sysplex and the same CEC; a dynamic IUTIQDIO link is created, whether ISTLSXCF is active or not.

For information about activating the ISTLSXCF major node, see z/OS Communications Server: SNA Resource Definition Reference.

Dynamic XCF can be enabled even in a single system sysplex. HiperSockets can be used between LPARs on the same central processor complex (CPC) even when MVS images in those LPARs are not defined to be part of the same sysplex. HiperSockets can also be used between LPARs even when some of those other LPARs are running Linux, as long as all of the stacks connecting to HiperSockets and needing to exchange IP packets with each other define IP addresses that are all in the same subnet (as defined by the dynamic XCF IP address and subnet mask in the IPCONFIG6 DYNAMICXCF profile statement).

A mix of static and dynamic IPv4 and IPv6 definitions for a device are not allowed. For example, if a static IUTSAMEH IPv4 device/link is defined, then the IPv6 dynamic definition for IUTSAMEH is not created. If a static IUTSAMEH IPv6 interface is defined, then the IPv4 dynamic definition for IUTSAMEH is not created. The same logic is also applied for XCF links; a mix of static and dynamic IPv4 and IPv6 definitions is not allowed for an XCF link.

ipv6_address

The fully qualified IPv6 address that is used for all dynamically generated XCF, Same Host, and HiperSockets interfaces.

See “Restrictions on IPv6 addresses configured in the TCP/IP profile” on page 143 for a list of restrictions that must be observed when specifying this parameter.

prefix_route_len

The length of the routing prefix (an integer value in the range 1 - 128). If specified, and if DYNAMICXCF generates a HiperSockets interface definition, TCP/IP creates a prefix route over the HiperSockets interface using the number of bits specified in *prefix_route_len* of the *ipv6_address*. Therefore, you can configure other stacks outside the sysplex for the same IQD CHPID using IP addresses with the same prefix such that this stack automatically has a route to these other stacks over the HiperSockets interface generated by DYNAMICXCF. If *prefix_route_len* is not specified, then TCP/IP does not create a prefix route over the HiperSockets interface. For interfaces other than HiperSockets which are generated from DYNAMICXCF, the *prefix_route_len* value has no meaning.

Guideline: Configure a *prefix_route_len* to simplify connectivity if you use HiperSockets on the same IQD CHPID for stacks outside the sysplex or if you configure VIPAROUTE statements.

INTFID *interface_id*

An optional 64-bit interface identifier in colon-hexadecimal format. IPv6 address shorthand notation (for example, the use of :: to indicate multiple groups of 16 bits of zeros) is not allowed when specifying the interface ID. If specified, this interface ID is used to form the link-local address for the interface.

If INTFID is not coded, TCP/IP generates a random value to be used to form the link-local address.

See “INTERFACE - IPAQENET6 OSA-Express QDIO interfaces statement” on page 161 for an explanation of restrictions that must be observed when manually specifying the INTFID parameter.

SOURCEVIPAINTERFACE *vipa_name*

The SOURCEVIPAINTERFACE parameter is optional. This parameter specifies which static VIPA interface is to be used as the source IP address when IPCONFIG6 SOURCEVIPA is specified and outbound packets are sent over the dynamically generated XCF or Same Host interfaces. The *vipa_name* value is the interface name for a VIRTUAL6 interface. If the VIPA has multiple IP addresses, then the source VIPA address for outbound packets is selected from among these addresses according to the default source address selection algorithm. The maximum length is 16 characters. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.

The use of the SOURCEVIPAINTERFACE parameter can be overridden. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

SECCLASS *security_class*

Use this parameter to associate a security class for IP filtering with each IPv6 dynamic XCF interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC6RULE statement in the TCP/IP profile.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

Restriction: This value is used only when IPSECURITY is specified on the IPCONFIG6 statement.

For more details about the use of DYNAMICXCF, see the DYNAMICXCF information in z/OS Communications Server: IP Configuration Guide. The DYNAMICXCF parameter is confirmed by the message:

```
EZZ0739I IPV6 DYNAMIC XCF DEFINITIONS ARE ENABLED
```

HOPLIMIT *hoplimit*

Number of hops a packet originating at this host can travel enroute to the destination. If the destination is more hops away, the packet never reaches the destination. The valid range is between 1 - 255. The default is 255.

ICMPERRORLIMIT *msgs_per_sec*

This parameter controls the rate at which ICMP error messages can be sent to a particular IPv6 destination address. The number specified is messages per second. The default is 3 messages per second, and the valid range is 1 - 20 messages per second. A token bucket algorithm is used to allow bursts of ICMP errors while limiting the long-term rate.

IGNOREREDIRECT

Causes TCP/IP to ignore ICMPv6 Redirect packets. The IGNOREREDIRECT parameter is confirmed by the message:

```
EZZ0701I ICMPV6 REDIRECTS WILL BE IGNORED
```

If you are using OMPROUTE, and IPv6 interfaces are configured to OMPROUTE, and this option is not specified, IGNOREREDIRECT is enabled automatically. If you are using intrusion detection services (IDS) policy to detect and discard ICMPv6 redirect packets and this option is not specified, ICMPv6 redirect packets are discarded while the policy is active.

IGNOREROUTERHOPLIMIT | NOIGNOREROUTERHOPLIMIT

NOIGNOREROUTERHOPLIMIT

NOIGNOREROUTERHOPLIMIT causes TCP/IP to not ignore a hop limit value received in a router advertisement from a router over an IPAQENET6 interface. This results in the configured global hop limit value being overridden by the router advertisement value for all routes using the interface on which the router advertisement was received. This is the default value. The NOIGNOREROUTERHOPLIMIT parameter is confirmed by the message:

```
EZZ0720I ROUTER ADVERTISEMENT HOP LIMIT VALUES WILL NOT BE IGNORED
```

IGNOREROUTERHOPLIMIT

Although you can configure a global hop limit value for the stack (by way of IPCONFIG6 HOPLIMIT), your stack might receive a different hop limit value in a router advertisement from a router, over an IPAQENET6 interface. This results in the configured global hop limit value being overridden by the router advertisement value for all routes using the interface on which the router advertisement was received. IGNOREROUTERHOPLIMIT gives you a way to prevent this, ensuring that your configured value is always used. The IGNOREROUTERHOPLIMIT parameter is confirmed by the message:

```
EZZ0719I ROUTER ADVERTISEMENT HOP LIMIT VALUES WILL BE IGNORED
```

IPSECURITY

Activates IPv6 IP filtering and IPv6 IPSec tunnel support. This parameter requires the IPSECURITY parameter to be configured for IPv4 on the IPCONFIG statement.

Requirements:

- Use this parameter so that the stack can function with the Communications Server IKE daemon and to enable the stack to receive IPv6 IPSec policy information, such as IP filter rules from the policy agent.
- Use this parameter so that the stack can receive IPv6 defensive filters from the Defense Manager daemon (DMD).

The IPSECURITY parameter is confirmed by the message:

```
EZZ0786I IPV6 SECURITY SUPPORT IS ENABLED
```

Restriction: IPSec functions can be activated only at initial activation of TCP/IP.

OSMSECCLASS *security_class*

Use this parameter to associate a security class for IP filtering with each OSM interface. In order for traffic over the interface to match a filter rule, the filter rule must have the same security class value as the interface or a value of 0. Filter rules can be specified in the TCP/IP profile or in an IP Security policy file read by the Policy Agent. Filter rules can include a security class specification on the IpService statement in an IP Security policy file or on the SECCLASS parameter on the IPSEC6RULE statement in the TCP/IP profile. For more information about OSM interfaces, see the TCP/IP in an intraensemble network section in z/OS Communications Server: IP Configuration Guide.

Valid security classes are identified as a number in the range 1 - 255. The default value is 255. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

MULTIPATH | NOMULTIPATH

NOMULTIPATH

Disables the multipath routing selection algorithm for outbound IP traffic. If there are multiple equal-cost routes to a destination and NOMULTIPATH is specified, TCP/IP uses the first active route found to send each IP packet. The NOMULTIPATH parameter is confirmed by the message:

```
EZZ0703I IPV6 MULTIPATH SUPPORT IS DISABLED
```

NOMULTIPATH is the default value.

Rule: The NOMULTIPATH parameter applies to outbound IP traffic that is routed by using the main route table. This parameter applies also to outbound IP traffic that is routed by using a policy-based route table if the Multipath6 UseGlobal parameter is specified on the RouteTable statement that defines the policy-based route table. See "RouteTable statement" on page 1152 for more information.

MULTIPATH

Enables the multipath routing selection algorithm for outbound IP traffic. In general, multipath routing provides the routing distribution necessary to balance the network utilization of outbound packets by load splitting. Multipath routing requires the definition of multiple equal-cost routes that are either defined statically or added dynamically by routing protocols (except for RIP, which does not provide multipath routing). If MULTIPATH is specified without any subparameters, the default is PERCONNECTION. The MULTIPATH parameter has no effect if there are no multipath routes in the TCP/IP configuration.

Rules:

- The MULTIPATH parameter and its subparameters apply to outbound IP traffic that is routed by using the main route table. This parameter and its subparameters apply also to outbound IP traffic that is routed by using a policy-based route table if the Multipath6 UseGlobal parameter is specified on the RouteTable statement that defines the policy-based route table. See "RouteTable statement" on page 1152 for more information.
- The multipath routing selection algorithm is applied and can be specified separately for each route table. Specify the algorithm for the main route table using the MULTIPATH parameter on the IPCONFIG6 statement. Specify the algorithm for policy-based route tables in the policy definition for each table. See "RouteTable statement" on page 1152 for more information.

Note: The IPCONFIG6 MULTIPATH|NOMULTIPATH configuration option affects Enterprise Extender (EE) traffic when MULTPATH=TCPVALUE is coded. For information about multipath for EE, see z/OS Communications Server: SNA Network Implementation Guide. For information about the MULTPATH start option, see z/OS Communications Server: SNA Resource Definition Reference.

PERCONNECTION

If there are multiple equal-cost routes to a destination and MULTIPATH PERCONNECTION is specified, TCP/IP, upon first sending an IP packet to a given destination, selects a route on a round-robin basis from a multipath routing list to that destination host. The selected route is used to route IP packets for a given connection or connectionless oriented association to that destination host. Connection or connectionless oriented IP packets using the same association always use the same route, as long as that route is active. The MULTIPATH PERCONNECTION parameter is confirmed by the message:

```
EZZ0704I IPV6 MULTIPATH PERCONNECTION SUPPORT IS ENABLED
```

PERPACKET

If there are multiple equal-cost routes to a destination, TCP/IP, upon sending an IP packet in that destination, selects a route on an approximate round-robin basis from a multipath routing list to that destination host. The selected route is used for routing that IP packet. Connection or connectionless oriented IP packets using the same source and destination address pair do not always use the same route, but do use all possible active routes to that destination host. All IP packets for a given association with a destination host are spread across the multiple equal-cost routes. The MULTIPATH PERPACKET parameter is confirmed by the message:

```
EZZ0704I IPV6 MULTIPATH PERPACKET SUPPORT IS ENABLED
```

Restriction: The MULTIPATH PERPACKET parameter cannot be enabled if the IPSECURITY parameter is specified. If both values are specified, the following messages are displayed, and multipath routing is disabled.

```
EZZ0792I CANNOT ENABLE IPV6 MULTIPATH PERPACKET SUPPORT WHEN
IPV6 SECURITY IS ENABLED
EZZ0703I IPV6 MULTIPATH SUPPORT IS DISABLED
```

SEGMENTATIONOFFLOAD | NOSEGMENTATIONOFFLOAD

Specifies whether the stack should offload TCP segmentation for IPv6 packets to OSA-Express features. The TCP segmentation offload support transfers the overhead of segmenting outbound data into individual TCP packets to QDIO-attached OSA-Express devices that support this function. Offloading segmentation of streaming-type workloads reduces CPU use and increases throughput. This parameter is ignored for OSA-Express features that do not support IPv6 segmentation offload.

See the steps for modifying topic for information about changing this parameter while the TCP/IP stack is active. See TCP segmentation offload in z/OS Communications Server: IP Configuration Guide for more information about the TCP segmentation offload support.

NOSEGMENTATIONOFFLOAD

TCP segmentation is performed by the TCP/IP stack. This value is the default value.

SEGMENTATIONOFFLOAD

TCP segmentation is offloaded to the OSA-Express feature.

SOURCEVIPA | NOSOURCEVIPA

NOSOURCEVIPA

Specifies that TCP/IP is not requested to use a VIPA address as the source IP address for outbound datagrams. The NOSOURCEVIPA parameter is confirmed by the message:

```
EZZ0702I IPV6 SOURCEVIPA SUPPORT IS DISABLED
```

NOSOURCEVIPA is the default value.

SOURCEVIPA

Requests that TCP/IP use a virtual IP address assigned to the TCPSTACKSOURCEVIPA interface (if TCPSTACKSOURCEVIPA is specified) or to the SOURCEVIPAINTERFACE interface as the source address for outbound datagrams that do not have an explicit source address. If multiple addresses are assigned to the TCPSTACKSOURCEVIPA interface or the SOURCEVIPAINTERFACE interface, the source address is selected from among these addresses according to the default source address selection algorithm. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.

Requirement: You must specify the SOURCEVIPAINTERFACE keyword on the INTERFACE statement for each interface on which you want that SOURCEVIPA to take effect. The SOURCEVIPA parameter is confirmed by the message:

```
EZZ0702I IPV6 SOURCEVIPA SUPPORT IS ENABLED
```

Tip: The use of SOURCEVIPA or TCPSTACKSOURCEVIPA can be overridden. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

TCPSTACKSOURCEVIPA | NOTCPSTACKSOURCEVIPA

NOTCPSTACKSOURCEVIPA
Specifies that TCP/IP does not use a stack-level IPv6 address as the source address for outbound TCP connections. The source IP address is determined by the normal default selection.

TCPSTACKSOURCEVIPA *intf_name*
The name of a static VIPA or a dynamic VIPA interface. The maximum length is 16 characters.

If the interface has multiple IP addresses, then the sourcevipa address for outbound packets is selected from among these addresses according to the default source address selection algorithm. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.

If SOURCEVIPA has not been enabled for IPCONFIG6, IPCONFIG6 TCPSTACKSOURCEVIPA is ignored and the following message is issued:

```
EZZ0760I IPV6 TCPSTACKSOURCEVIPA IS IGNORED - SOURCEVIPA IS NOT ENABLED
```

Tips:

- After it is set, TCPSTACKSOURCEVIPA is not disabled until a profile explicitly adds NOTCPSTACKSOURCEVIPA to the IPCONFIG6 statement.
- A dynamic VIPA that becomes inactive because it moves to another TCP/IP stack, or that is deleted because the application that caused its creation (in the case of a VIPARANGE statement created address) causes its deletion, is no longer a valid interface for the TCPSTACKSOURCEVIPA parameter.
- If you specify the same distributed DVIPA interface for TCPSTACKSOURCEVIPA on multiple target stacks, you also should specify SYSPLEXPORTS on the VIPADISTRIBUTE statement. Otherwise connections might be disrupted because identical connections could be created from more than one stack.
- Carefully consider the following when determining the interface to use for TCPSTACKSOURCEVIPA.
  - A dynamic VIPA that becomes inactive because it moves to another TCP/IP stack, or that is deleted because the application

that caused its creation (in the case of a VIPARANGE statement created address) causes its deletion, is no longer a valid interface for TCPSTACKSOURCEVIPA.
  - A dynamic VIPA interface that is created by a VIPARANGE statement can have multiple dynamic VIPA addresses associated with it. The actual address chosen as the source IP for the outbound connection is not predictable or necessarily meaningful.
- The use of TCPSTACKSOURCEVIPA can be overridden. See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined.

TEMPADDRS | NOTEMPADDRS

NOTEMPADDRS

Specifies that TCP/IP should not generate IPv6 temporary addresses. Use of the NOTEMPADDRS parameter is confirmed by the message:

```
EZZ0821I IPV6 TEMPORARY ADDRESS SUPPORT IS DISABLED
```

NOTEMPADDRS is the default value.

TEMPADDRS

Requests that TCP/IP generate IPv6 temporary addresses for IPAQENET6 OSA-Express QDIO interfaces for which stateless address autoconfiguration is enabled. Stateless address autoconfiguration is enabled for an interface if no address or prefix is specified with the IPADDR keyword. See the information about using IPv6 temporary addresses to address privacy concerns in the z/OS Communications Server: IPv6 Network and Application Design Guide.

Requirement: You must specify the job name of an application in the SRCIP statement block with a value of TEMPADDRS to cause a temporary IPv6 address to be preferred over a public IPv6 address as the source IP address for the application; otherwise, the default source address selection algorithm prefers public IPv6 addresses over temporary addresses. See the information about default source address selection in the z/OS Communications Server: IPv6 Network and Application Design Guide .

The TEMPADDRS parameter is confirmed by the message:

```
EZZ0816I IPV6 TEMPORARY ADDRESS SUPPORT IS ENABLED
```

PREFLIFETIME *pref_lifetime*

Preferred lifetime for temporary addresses specified in hours. At the expiration of the preferred lifetime, a new temporary address is generated and the existing address is deprecated. Valid values are in the range 1 - 720 hours (30 days). The default is 24 hours (1 day).

Results:

- A temporary address can be deprecated sooner than specified by the *pref_lifetime* value if the preferred lifetime of the prefix that is learned from a router advertisement is less than the *pref_lifetime*.
- A short preferred lifetime results in new temporary addresses being generated more quickly.

VALIDLIFETIME *valid_lifetime*

Valid lifetime for temporary addresses, specified in hours. At the expiration of the valid lifetime, the temporary address is deleted. Valid values are in the range 2 - 2160 hours (90 days). The default is 7 times the preferred lifetime, not to exceed a maximum value of 90 days.

Rules:

- *valid_lifetime* value must be greater than *pref_lifetime* value.
- If PREFLIFETIME is not explicitly configured, the *valid_lifetime* value must be greater than the default value for *pref_lifetime*.

Results:

- A temporary address can be deleted sooner than specified by the *valid_lifetime* value if the valid lifetime of the prefix that is learned from a router advertisement is less than *valid_lifetime*.
- A short valid lifetime results in deprecated temporary addresses being deleted more quickly.

Guideline: Do not specify a small *pref_lifetime* value with a large *valid_lifetime* value. A large number of deprecated temporary addresses can have an impact on storage usage.

default_valid_lifetime
Specifies the default valid lifetime for temporary addresses in hours. The default is 7 times the preferred lifetime; you can specify a maximum value of 90 days.

Steps for modifying

To modify most parameters for the IPCONFIG6 statement, you must respecify the statement with the new parameters. Additional actions are required to modify the following parameters:

CHECKSUMOFFLOAD | NOCHECKSUMOFFLOAD
If the CHECKSUMOFFLOAD or NOCHECKSUMOFFLOAD parameter is changed with the VARY TCPIP,,OBEYFILE command, the new value does not affect any active OSA-Express QDIO interfaces. For this change to affect an active OSA-Express QDIO interface, the interface must be stopped and restarted.

DYNAMICXCF
None of the parameters on the IPCONFIG6 DYNAMICXCF statement can be changed with a VARY TCPIP,,OBEYFILE command. You must first stop the TCP/IP stack, apply changes, and then restart the TCP/IP stack.

If dynamic XCF definitions have been enabled but a later VARY TCPIP,,OBEYFILE command contains NODYNAMICXCF, only future dynamic definitions and connectivity are affected. Existing definitions and connectivity are not affected.

IPSECURITY
z/OS IPv6 IPSec functions cannot be activated using the VARY TCPIP,,OBEYFILE command on an active TCP/IP stack. To activate z/OS IPSec for IPv6, halt all traffic on the designated TCP/IP stack, stop the stack, modify the TCP profile to include IPCONFIG6 IPSECURITY, and restart the stack.

MULTIPATH
If you modify the multipath routing type (PERCONNECTION to PERPACKET, or vice versa), the new parameter takes effect only for new connections created after the modification is done, and existing connections

use whatever the value was when the connection was established. If you enable multipath routing when it was previously disabled, existing connections are not affected; multipath routing is applied to new connections only.

SEGMENTATIONOFFLOAD | NOSEGMENTATIONOFFLOAD
If the SEGMENTATIONOFFLOAD or NOSEGMENTATIONOFFLOAD parameter is changed with the VARY TCPIP,,OBEYFILE command, the new value does not affect any active OSA-Express QDIO interfaces. For this change to affect an active OSA-Express QDIO interface, the interface must be stopped and restarted.

TEMPADDRS
If you disable temporary addresses by changing TEMPADDRS to NOTEMPADDRS using a VARY TCPIP,,OBEYFILE command, all existing IPv6 temporary addresses are deleted. This is disruptive for connections that are using the temporary address.

Related topics

- “GLOBALCONFIG statement” on page 117
- “IPCONFIG statement” on page 190
- “SRCIP statement” on page 282

IPSEC statement

Use the IPSEC statement to define policy for the IPv4 security function that is enabled with the IPCONFIG IPSECURITY parameter. The IPSEC statement is ignored if IPSECURITY is not specified on the IPCONFIG statement. If you also enable IPv6 Security with the IPCONFIG6 IPSECURITY parameter, then use the IPSEC statement to also define policy for IPv6 IP security.

Restriction: Only one IPSEC statement block should appear in the profile. Any subsequent statement blocks are ignored and an informational message is generated. Multiple filter rules can be defined in the IPSEC block.

Syntax

Rule: Specify the parameters in the order shown here.

```
               ┌─────────────────┐   ┌────────────────────┐
               ▼                 │   ▼                    │
►►──IPSEC──────┬─────────────────┴───────┤ IP Filter Rule ├──ENDIPSEC──────►◄
               ├─DVIPsec────────┤
               │ ┌─LOGDISable─┐ │
               ├─┴────────────┴─┤
               │ └─LOGENable──┘ │
               │ ┌─NOLOGImplicit─┐│
               └─┴───────────────┴┘
                 └─LOGImplicit───┘
```

IP Filter Rule:

```
├──┬─IPv4 Filter Rule─┬──────────────────────────────────────────────────┤
   └─IPv6 Filter Rule─┘
```

IPv4 Filter Rule:

```
├──IPSECRule──┬─src_ipaddr───────────────┬──┬─dest_ipaddr───────────────┬──────────►
              ├─src_ipaddr/prefix_length─┤  ├─dest_ipaddr/prefix_length─┤
              └─*────────────────────────┘  └─*─────────────────────────┘

    ┌─NOLOG─┐                 ┌─ROUTING LOCAL──────────┐  ┌─SECCLASS 0─────────────┐
►───┼───────┼──┤ Protocol ├───┼────────────────────────┼──┼────────────────────────┼──┤
    └─LOG───┘                 └─ROUTING──┬─ROUTED─┬────┘  └─SECCLASS securityclass─┘
                                         └─EITHER─┘
```

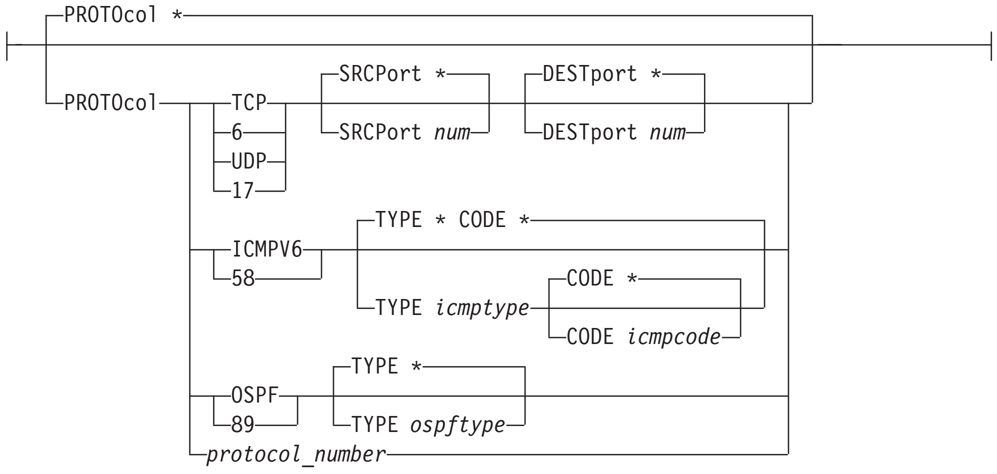
Protocol:

```
   ┌─PROTOcol *──────────────────────────────────────────────────────┐
├──┼─────────────────────────────────────────────────────────────────┼──────────┤
   │                     ┌─SRCPort *───┐  ┌─DESTport *───┐           │
   └─PROTOcol──┬─┬─TCP─┬─┼─────────────┼──┼──────────────┼─────┬─────┘
               │ ├─6───┤ └─SRCPort num─┘  └─DESTport num─┘     │
               │ ├─UDP─┤                                       │
               │ └─17──┘                                       │
               │         ┌─TYPE * CODE *───────────────────┐   │
               ├─┬─ICMP─┬┼─────────────────────────────────┼───┤
               │ └─1────┘│               ┌─CODE *────────┐ │   │
               │         └─TYPE icmptype─┼───────────────┼─┘   │
               │                         └─CODE icmpcode─┘     │
               │         ┌─TYPE *────────┐                     │
               ├─┬─OSPF─┬┼───────────────┼─────────────────────┤
               │ └─89───┘└─TYPE ospftype─┘                     │
               └─protocol_number───────────────────────────────┘
```

IPv6 Filter Rule:

```
├──IPSEC6Rule──┬─src_ipaddr───────────────┬──┬─dest_ipaddr───────────────┬─────────►
               ├─src_ipaddr/prefix_length─┤  ├─dest_ipaddr/prefix_length─┤
               └─*────────────────────────┘  └─*─────────────────────────┘

    ┌─NOLOG─┐                 ┌─ROUTING LOCAL──────────┐  ┌─SECCLASS 0─────────────┐
►───┼───────┼──┤ Protocol ├───┼────────────────────────┼──┼────────────────────────┼──┤
    └─LOG───┘                 └─ROUTING──┬─ROUTED─┬────┘  └─SECCLASS securityclass─┘
                                         └─EITHER─┘
```

Protocol:



Parameters

DVIPSEC

Indicates that IPsec tunnels associated with IPv4 dynamic VIPA addresses are eligible to be distributed if the dynamic VIPA address is being distributed. The IPsec tunnels are also eligible to be moved during dynamic VIPA takeover or giveback.

Restriction: For tunnels that traverse a NAT device, the dynamic VIPA takeover and giveback function is limited to configurations where IKE can act as initiator. IKE cannot act as initiator in the following configurations:

- The remote security endpoint is a security gateway and a NAT is being traversed
- The remote security endpoint is behind an NAPT

For more information about NAT Traversal configuration scenarios, see z/OS Communications Server: IP Configuration Guide.

LOGDISABLE/LOGENABLE

Indicates whether packet filter logging is enabled or disabled. The following log messages are controlled by this parameter:

- EZD0814I
- EZD0815I
- EZD0821I
- EZD0832I
- EZD0833I
- EZD0836I
- EZD0822I

If logging is enabled, messages are written to syslogd by the Traffic Regulation Manager Daemon (TRMD).

If LOGENABLE is specified, then the log setting on the individual default filter rules and the implicit default rules is honored. The log setting for individual default rules is specified with the LOG/NOLOG parameter. The log setting for the implicit default rules is specified with the LOGIMPLICIT/ NOLOGIMPLICIT parameter.

If LOGDISABLE is specified, then the log setting on the individual default filter rules and the implicit default rules is ignored and no packet filter logging is done.

LOGIMPLICIT/NOLOGIMPLICIT

Indicates whether packet filter logging is enabled or disabled for packets that are denied by the implicit default rules. IP traffic not explicitly permitted by the default IP filter rules parameters described in the following IP Filter Rule parameters topic, is handled by implicit default rules generated by the stack while default IP filter policy is in effect.

If the IPSEC statement is not specified, packet filter logging is disabled for packets that are handled by the implicit default rules. To turn on packet filter logging for the implicit default rules, IPSEC must be coded with the LOGENABLE and LOGIMPLICIT parameters.

A setting of LOGIMPLICIT is honored only when filter logging is enabled on the IPSEC statement with LOGENABLE.

IP Filter Rule parameters

Default IP filter rules can be defined on the IPSEC statement. The default IP filter policy is used prior to the initial loading of IP security policy into the stack from the Policy Agent. It is also used when the IP security policy has been suspended by the z/OS UNIX ipsec command (that is, when the ipsec -f default command is issued).

The default IP filter policy consists of the following rules:

- Rules defined explicitly with the IPSECRULE and IPSEC6RULE statement
- Implicit rules that deny all inbound and outbound data traffic

The explicit rules appear first in the search order and the implicit deny all rules appear last in the search order.

The rules defined explicitly with the IPSECRULE and IPSEC6RULE statements are permit rules. Each rule is treated as bidirectional, generating both an outbound and inbound permit rule. The outbound rule permits outbound traffic from the specified source to the specified destination. The inbound rule permits inbound traffic with the destination and source reversed. IP traffic not explicitly permitted by one of the defined rules is denied while the default IP filter policy is in effect.

The physical order in which the rules are defined in the profile determines the search order for the rules. The rule parameters are ANDed together to determine whether the IP traffic matches the filter rule.

If you configure an IPSEC6RULE statement but did not specify IPCONFIG6 IPSECURITY, then TCP/IP rejects the IPSEC6RULE statement and issues message EZZ0787I in z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM).

If the IPSEC statement is not specified or if no default IP filter rules are specified, the default IP filter table consists only of the implicitly defined **deny all** rule.

src_ipaddr

The source IP address for the outbound rule. For outbound IP traffic to be permitted by this rule, the source IP address of the traffic must match this parameter. For inbound IP traffic to be permitted by the generated inbound rule, the destination IP address of the traffic must match this parameter.

Specify an asterisk (*) to allow any source IP address to match.

Guidelines:

- For IPSECRULE, an asterisk means any IPv4 address. For IPSEC6RULE, an asterisk means any IPv6 address
- For IPSEC6RULE, the *src_ipaddr* can be any valid IPv6 address in colon-hexadecimal format. IPv4-mapped IPv6 addresses are also allowed.

src_ipaddr/prefix-length

A source IP prefix specification for the outbound rule. For outbound IP traffic to be permitted by this rule, the leading portion of the source IP address of the traffic must match the leading portion of the source IP address (*src_ipaddr*) rule for the number of bits indicated by the *prefix-length* value. For inbound IP traffic to be permitted by the generated inbound rule, the destination IP address of the traffic must match the leading portion of the source IP address (*src_ipaddr*) rule for the number of bits indicated by the *prefix-length* value. For IPSECRULE, the *prefix-length* is a value in the range 1 - 32. For IPSEC6RULE, the *prefix-length* is a value in the range 1 - 128.

dest_ipaddr

The destination IP address for the outbound rule. For outbound IP traffic to be permitted by this rule, the destination IP address of the traffic must match this parameter. For inbound IP traffic to be permitted by the generated inbound rule, the source IP address of the traffic must match this parameter.

Specify an asterisk (*) to allow any destination IP address to match.

Guidelines:

- For IPSECRULE, an asterisk means any IPv4 address. For IPSEC6RULE, an asterisk means any IPv6 address
- For IPSEC6RULE, the *dst_ipadd* can be any valid IPv6 address in colon-hexadecimal format. IPv4-mapped IPv6 addresses are also allowed.

dest_ipaddr/prefix-length

A destination IP prefix specification for the outbound rule. For outbound IP traffic to be permitted by this rule, the leading portion of the destination IP address of the traffic must match the leading portion of the destination address (*dest_ipaddr*) rule for the number of bits indicated by the *prefix-length* value. For inbound IP traffic to be permitted by the generated inbound rule, the source IP address of the traffic must match the leading portion of the destination address (*dest_ipaddr*) rule for the number of bits indicated by the *prefix-length* value. For IPSECRULE, the *prefix-length* is a value in the range 1 - 32. For IPSEC6RULE, the *prefix-length* is a value in the range 1 - 128.

LOG/NOLOG

Indicates whether packet filter logging is enabled or disabled for the default filter rule. A setting of LOG is honored only when filter logging is enabled on the IPSEC statement with LOGENABLE.

PROTOCOL

The protocol specification for this rule. For IP traffic to be permitted by this rule, the protocol of the traffic must match this parameter.

* Any protocol specification. IP traffic of any protocol can match this rule. This is the default value.

TCP | 6
TCP protocol specification. For IP traffic to be permitted by this rule, the protocol of the traffic must be TCP.

SRCPORT *num*
A source port specification for the outbound rule. The parameter is applicable when either TCP or UDP is specified for PROTOCOL. For outbound IP traffic to be permitted by this rule, the source port of the traffic must match this parameter. For inbound IP traffic to be permitted by the generated inbound rule, the destination port of the traffic must match this parameter.

Valid values for *num* are in the range 1 - 65535. The default is an asterisk (*), which indicates that any source port matches this parameter.

Rule: If the ROUTING value is ROUTED or EITHER, SRCPORT must be defined as all ports (*).

DESTPORT *num*
A destination port specification for the outbound rule. The parameter is applicable when either TCP or UDP is specified for PROTOCOL. For outbound IP traffic to be permitted by this rule, the destination port of the traffic must match this parameter. For inbound IP traffic to be permitted by the generated inbound rule, the source port of the traffic must match this parameter.

Valid values for *num* are in the range 1 - 65535. The default is *, which indicates that any destination port matches.

Restriction: If the ROUTING value is ROUTED or EITHER, DESTPORT must be defined as all ports (*).

UDP | 17
UDP protocol specification. For IP traffic to be permitted by this rule, the protocol of the traffic must be UDP.

SRCPORT *num*
A source port specification for the outbound rule. The parameter is applicable when either TCP or UDP is specified for PROTOCOL. For outbound IP traffic to be permitted by this rule, the source port of the traffic must match this parameter. For inbound IP traffic to be permitted by the generated inbound rule, the destination port of the traffic must match this parameter.

Valid values for *num* are in the range 1 - 65535. The default is *, which indicates that any source port matches.

Restriction: If the ROUTING value is ROUTED or EITHER, SRCPORT must be defined as all ports (*).

DESTPORT *num*

A destination port specification for the outbound rule. The parameter is applicable when either TCP or UDP is specified for PROTOCOL. For outbound IP traffic to be permitted by this rule, the destination port of the traffic must match this parameter. For inbound IP traffic to be permitted by the generated inbound rule, the source port of the traffic must match this parameter.

Valid values for *num* are in the range 1 - 65535. The default is *, which indicates that any destination port matches.

Restriction: If the ROUTING value is ROUTED or EITHER, DESTPORT must be defined as all ports (*).

ICMP | 1

ICMP protocol specification.

Restrictions:

- The ICMP protocol is valid only on an IPSECRULE statement.
- For IP traffic to be permitted by this rule, the protocol of the traffic must be ICMP.

TYPE *icmptype*

ICMP type. This parameter is applicable when ICMP is specified for the PROTOCOL parameter. Valid values are an asterisk (*) or are in the range 0 - 255. The default is *, which indicates that any ICMP type matches.

Restrictions:

- For IP traffic to be permitted by this rule, the ICMP type of the traffic must match this parameter value.
- If the ROUTING value is ROUTED or EITHER, TYPE must be defined as all types (*).

CODE *icmpcode*

ICMP code. This parameter is applicable when ICMP is specified for the PROTOCOL parameter and when the TYPE parameter has a value other than an asterisk (*) for the *icmptype*. Valid values are asterisk (*) or in the range 0 - 255. The default is asterisk (*), which indicates that any ICMP code matches.

Restrictions:

- For IP traffic to be permitted by this rule, the ICMP code of the traffic must match this parameter value.
- If the ROUTING value is ROUTED or EITHER, CODE must be defined as all codes (*).

ICMPV6 | 58

ICMPv6 protocol specification.

Restriction: The ICMPv6 protocol is valid only on an IPSEC6RULE statement.

Rule: For IP traffic to be permitted by this rule, the protocol of the traffic must be ICMPv6.

TYPE *icmptype*

ICMP type. This parameter is applicable when ICMPV6 is specified for PROTOCOL. Valid values are * or 0 - 255. The default is *, which indicates that any ICMP type matches.

Restrictions:

- For IP traffic to be permitted by this rule, the ICMP type of the traffic must match this parameter value.
- If the ROUTING value is ROUTED or EITHER, TYPE must be defined as all types (*).

CODE *icmpcode*

ICMP code. This parameter is applicable when ICMPV6 is specified for PROTOCOL and when TYPE has been specified with an *icmptype* value other than *. Valid values are * or 0 - 255. The default is *, which indicates that any ICMP code matches.

Restrictions:

- For IP traffic to be permitted by this rule, the ICMP code of the traffic must match this parameter value.
- If the ROUTING value is ROUTED or EITHER, CODE must be defined as all codes(*).

OSPF | 89

OSPF protocol specification.

Restriction: For IP traffic to be permitted by this rule, the protocol of the traffic must be OSPF.

TYPE *ospftype*

OSPF type. This parameter is applicable when OSPF is specified for PROTOCOL. Valid values are * or 0 - 255.

Restrictions:

- For IP traffic to be permitted by this rule, the OSPF type of the traffic must match this parameter value. The default is *, which indicates that any OSPF type matches.
- If the ROUTING value is ROUTED or EITHER, TYPE must be defined as all types(*).

For a list of the possible IPv4 OSPF types, see RFC 1583 OSPF Version 2. For a list of the possible IPv6 OSPF types, see RFC 2740, OSPF for IPv6. See Appendix C, "Related protocol specifications," on page 1465 for more information about accessing RFCs.

protocol_number

A protocol number in the range 0 - 255.

Restriction: For IP traffic to be permitted by this rule, the protocol of the traffic must match this parameter.

ROUTING
Specifies the type of packet to which this rule applies. Valid values for ROUTING are:

LOCAL
Indicates that this rule applies to packets destined for this stack.

ROUTED
Indicates that this rule applies to packets being forwarded by this stack.

EITHER
Indicates that this rule applies to forwarded and non-forwarded packets.

The default value is LOCAL.

SECCLASS *security_class*
A security class value in the range 0 - 255.

Restriction: For IP traffic to be permitted by this rule, the security class of the interface that the traffic is inbound to or outbound from must match this parameter.For IPv4, the security class for the interface is specified as SECCLASS on the LINK, INTERFACE, or IPCONFIG DYNAMICXCF statement. For IPv6, the security class for the interface is specified as SECCLASS on the INTERFACE or IPCONFIG6 DYNAMICXCF statement. A value of 0 matches any security class value coded on the corresponding profile statement which defines the interface. For more information about security class values, see z/OS Communications Server: IP Configuration Guide.

The default value is 0.

Steps for modifying

To modify most parameters for the IPSEC statement, use a VARY TCPIP,,OBEYFILE command with a data set that contains a new IPSEC statement. Additional actions are required to modify the following parameters:

DVIPSEC
The value of DVIPSEC cannot be modified using the VARY TCPIP,,OBEYFILE command on an active TCP/IP stack.

LOGDISABLE/LOGENABLE
The value of LOGDISABLE/LOGENABLE can be modified using a VARY TCPIP,,OBEYFILE command with a data set that contains a new IPSEC statement. The current set of IPSECRULE statements should be included in the data set when changing LOGDISABLE/LOGENABLE on the IPSEC statement.

LOGIMPLICIT/NOLOGIMPLICIT
The value of LOGIMPLICIT/NOLOGIMPLICIT can be modified using a VARY TCPIP,,OBEYFILE command with a data set that contains a new IPSEC statement. The current set of IPSECRULE statements should be included in the data set when changing LOGIMPLICIT/NOLOGIMPLICIT on the IPSEC statement.

IP Filter Rules
To modify the default IP filter rules on the IPSEC statement, use a VARY TCPIP,,OBEYFILE command with a data set that contains a new IPSEC

statement. All existing default IP filter rules are deleted and replaced with the default IP filter rules defined on the new IPSEC statement.

To delete all defined default filter rules leaving only the implicit deny all default rule, the data set must contain a new IPSEC statement with no default filter rules defined. If the data set does not contain an IPSEC statement, then the existing default filter rules remain in effect.

If IP filtering is being done based on the default filter rules, then the modified default filter rules are in effect following the VARY TCPIP,,OBEYFILE command. If IP filtering is being done based on the filter rules defined to Policy Agent, then the default filter rules are updated by the VARY TCPIP,,OBEYFILE command, but filter rules defined in Policy Agent remain in effect. The ipsec -f default command must be issued to cause the default filter rules to be used.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands.

Examples

```
IPSEC
; Rule   SourceIp      DestIp    Logging   Prot       SrcPort    DestPort   Routing    Secclass
;
; Permit outbound IPv4 TCP traffic from local IP address 1.1.1.1 port 23 to remote IP address 2.2.2.2
; Permit inbound IPv4 TCP  traffic from remote IP address 2.2.2.2 to local IP address 1.1.1.1 port 23
  IPSECR 1.1.1.1       2.2.2.2   NOLOG     PROTO TCP  SRCPORT 23 DESTPORT * ROUTING LOCAL
;
; Permit outbound IPv4 TCP traffic from local IP address 1.1.1.1 to remote IP address 2.2.2.2 port 23
; Permit inbound IPv4 TCP  traffic from remote IP address 2.2.2.2 port 23 to local IP address 1.1.1.1
  IPSECR 1.1.1.1       2.2.2.2   NOLOG     PROTO TCP  SRCPORT *  DESTPORT 23
;
; Permit outbound IPv4 ICMP traffic from local IP addresses 1.2.0.0/16
; Permit inbound IPv4 ICMP  traffic to local IP addresses 1.2.0.0/16
  IPSECR 1.2.0.0/16    *         LOG       PROTO ICMP
; Permit all routed IPv4 traffic
; IPSECR *             *         LOG       PROTO *                         ROUTING ROUTED
; Permit all local outbound traffic to remote IP address 1.2.3.4
; Permit all local inbound  traffic from remote IP address 1.2.3.4
  IPSECR *             1.2.3.4
; Permit local outbound IPv6 Neighbor Solicitations
; Permit local inbound IPv6 Neighbor Solicitations
 IPSEC6R *             *                   LOG       PROTO  ICMPV6  TYPE 135
; Permit local outbound IPv6 Neighbor Advertisements
; Permit local inbound IPv6 Neighbor Advertisements
  IPSEC6R *            *                   LOG       PROTO  ICMPV6  TYPE  136
; Permit local inbound IPv6 Router Advertisements from remote IP address 2001::1:2:3:4
  IPSEC6R *            2001::1:2:3:4/128     LOG      PROTO  ICMPV6  TYPE 134

ENDIPSEC
```

Related topics

- “Summary of DEVICE and LINK statements” on page 47
- “Summary of INTERFACE statements” on page 141
- “IPCONFIG statement” on page 190
- “IPCONFIG6 statement” on page 206

ITRACE statement

Use the ITRACE statement to control TCP/IP runtime tracing. This statement is used primarily for diagnostic purposes.

Syntax

Rule: Specify the parameters in the order shown here.

```
                    ┌─SUBAGENt 1 CONFig 1 COMMAND 1 AUTODAEMON─┐
►►──ITRACE───┬─ON──┼─────────────────────────────────────────┼──┬──────────────►◄
             │      ├─SUBAGENt level──────────────────────────┤  │
             │      ├─CONFig level────────────────────────────┤  │
             │      ├─COMMAND level───────────────────────────┤  │
             │      └─AUTODAEMON──────────────────────────────┘  │
             │        ┌─SUBAGENt CONFig COMMAND AUTODAEMON─┐     │
             └─OFF───┼─────────────────────────────────────┼─────┘
                      ├─SUBAGENt────────────────────────────┤
                      ├─CONFig──────────────────────────────┤
                      ├─COMMAND─────────────────────────────┤
                      └─AUTODAEMON──────────────────────────┘
```

Parameters

ON Specify ON to establish runtime tracing. If specified with no parameters, the trace defaults to CONFIG level 1, SUBAGENT level 1, COMMAND level 1, and AUTODAEMON tracing.

OFF
Specify OFF to terminate runtime tracing. If specified with no parameters, CONFIG, SUBAGENT, COMMAND, and AUTODAEMON tracing is turned off.

SUBAGENT
Turn internal trace for SNMP subagent ON or OFF.

CONFIG
Turn internal trace for configuration ON or OFF.

Restrictions:

- To trace the processing of specific TCP/IP profile statements, you must specify the ITRACE statement before the profile statements in the profile data set.
- You cannot specify the ITRACE statement inside a block statement. For example, do not place it within a VIPADYNAMIC/ENDVIPADYNAMIC block or a BEGINROUTES/ENDROUTES block.

COMMAND
Turn internal trace for command ON or OFF.

AUTODAEMON
Turn internal trace for the autolog subtask ON or OFF.

level
Indicates the tracing level to be established. Levels are as follows:

Levels for CONFIG

1	ITRACE for all of config
2	General level of tracing for all of config
3	Tracing for configuration set commands
4	Tracing for configuration get commands
5	Tracing for syslog calls issued by config

100	Tracing for the parser
200	Tracing for scanner
300	Tracing for mainloop
400	Tracing for commands

Levels for SUBAGENT

1	General subagent tracing
2	General subagent tracing plus DPI traces
3	General subagent tracing plus extended storage dump traces
4	All trace levels

Levels for COMMAND

1	ITRACE for all commands

Steps for modifying

To modify parameters for the ITRACE statement, use a VARY TCPIP,,OBEYFILE command with a data set that contains a new ITRACE statement.

Examples

```
ITRACE ON CONFIG 3
ITRACE OFF SUBAGENT
```

Results:
- Subagent trace output is directed to the syslog daemon. This daemon is configured by the */etc/syslog.conf* z/OS UNIX file and must be active.
- AUTOLOG trace output goes to the destination specified by the ALGPRINT setting in the TCP/IP cataloged procedure (TCPIPROC).
- CONFIG trace output goes to the destination specified by the CFGPRINT setting in the TCP/IP cataloged procedure (TCPIPROC). If CFGPRINT is not specified in TCPIPROC, the CONFIG component dynamically allocates a ddname of CFGPRINT.
- Command trace output goes to the hardcopy console log.

Usage notes

ITRACE ON commands are cumulative until an ITRACE OFF is issued.

Related topics
- z/OS Communications Server: IP Diagnosis Guide
- "Specifying TCP/IP address space parameters" on page 343

NETACCESS statement

Use the NETACCESS statement to configure network access control. Specifically, it allows for the one-to-one mapping between a network, subnetwork or host and a Security Access Facility (SAF) resource name. The network specifications are used to build an internal data structure that maps networks, subnetworks and hosts to SAF resource names. The mapping is used to construct a complete resource name that is passed to the Security Product to determine the user's permission to access the network resource. The most specific mapping is used to determine the resource name for the SAF authorization check.

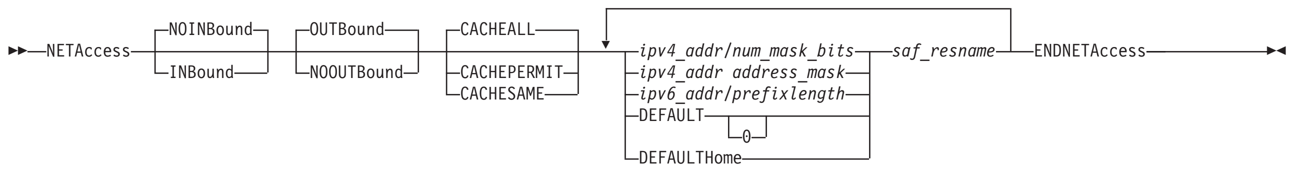
If the network resource does not have an assigned mapping, no SAF check is performed. If the network resource does have an assigned mapping, the SERVAUTH class must be active, the resource name must be defined, and the user ID making the request must have at least read access to the resource.

Inbound socket commands include application requests to bind a socket, accept a TCP connection and any command that transfers data into the application from a socket. Outbound socket commands include application requests to connect a socket and any command that transfers data from the application into the socket.

Multilevel-security is an enhanced security environment that can be configured on a z/OS Communications Server system. In this environment the Security Server and trusted resource managers enforce mandatory access control (MAC) policies in addition to the usual discretionary access control (DAC) policies. For more information about the multilevel-security environment and configuring z/OS Communications Server in that environment, see the multilevel-security information in the z/OS Communications Server: IP Configuration Guide.

Syntax

Rule: Specify the parameters in the order shown here.



Parameters

NOINBOUND
: Specifies that network access control checking is disabled for inbound socket commands. This is the default value.

INBOUND
: Specifies that network access control checking is enabled for inbound socket commands.

OUTBOUND
: Specifies that network access control checking is enabled for outbound socket commands. This is the default value.

NOOUTBOUND
: Specifies that network access control checking is disabled for outbound socket commands.

CACHEALL
: Specifies that when a SAF call is made to check a user's access to a security zone, the result is cached regardless of whether access is permitted or denied. Subsequent checks of the user's access to the security zone are resolved using the cached results. This is the default value.

This parameter allows an external security manager to write an audit record for only the first access check made for a user for each security zone.

CACHEPERMIT
Specifies that when a SAF call is made to check a user's access to a security zone, the result is cached when access is permitted, but not when access is denied. Subsequent checks of the user's access to a permitted security zone are resolved using the cached results. Subsequent checks of the user's access to a denied security zone are resolved by another SAF call.

This parameter allows an external security manager to write an audit record for only the first access check made for a user for each permitted security zone, and for all access checks made for a user for each denied security zone.

CACHESAME
Specifies that when a SAF call is made to check the access of a user to a security zone, the result is cached when access is permitted, but not when access is denied.

If the user is permitted to access the security zone, subsequent checks of the user access to the security zone are resolved using the cached results as long as the user associated with the socket and the IP address being accessed are unchanged. However, if the user that is associated with the socket changes or if the IP address being accessed changes from the previous packet that is received or sent over the socket, the next access check is resolved by another SAF call.

Subsequent checks of the user access to a denied security zone are resolved by another SAF call.

This parameter allows an external security manager to write an audit record for all denied access checks that are made for a user for each denied security zone and for the first of multiple successive access checks made for a socket under the same user and for the same IP address in a permitted security zone.

ipv4_addr/num_mask_bits
Specifies the network for which security product access control is required for user requests. The *num_mask_bits* field is used to create an address mask that is bit-contiguous from left to right. This address mask is logically ANDed with the *ipv4_addr* value to create the network address for which access control is required.

ipv4_addr address_mask
Specifies the network for which security product access control of user requests is required. The *address_mask* value is a bit mask (expressed in dotted decimal form) that is bit-contiguous from left to right. The *address_mask* value is logically ANDed with the *ipv4_addr* value to create the network address for which access control is required.

ipv6_addr/prefixlength
Specifies the IPv6 network for which security product access control is required. The *ipv6_addr* is an IPv6 address in colon-hexadecimal format. The *prefixlength* value is a decimal value specifying how many of the leftmost contiguous bits of the address comprise the prefix. The value is in the range of 1 - 128. IPv4-mapped IPv6 addresses and IPv6 addresses with the reserved prefix ::/96 are not allowed.

DEFAULT
Specifies that security product access control of user requests is required for any networks not specifically defined by other NETACCESS statement entries. If DEFAULTHOME is not specified, DEFAULT maps all addresses, local and

remote, not mapped by other entries. If DEFAULTHOME is also specified, DEFAULT maps all remote addresses not mapped by other entries. Use of the address_mask value of 0 on this entry is deprecated.

DEFAULTHOME
Specifies that security product access control of user requests is required for all IP addresses that are local to this stack and not specifically defined by other NETACCESS statement entries. When this parameter is specified, security product access control of user requests is also required for addresses dynamically defined by SYSPLEX services and IPv6 link-local and global addresses that are automatically assigned for an interface.

saf_resname
Specifies the final qualifier of a security product resource name. The maximum length is eight characters. The profile name has the following format:

```
EZB.NETACCESS.sysname.tcpname.saf_resname
```

where

- EZB.NETACCESS is constant.
- *sysname* is the value of the MVS &SYSNAME. system symbol.
- *tcpname* is the name of the procedure used to start the TCP stack.
- *saf_resname* is the 1-8 character value following the network specification.

If the installation's SAF compliant security product (for example, RACF) supports the SERVAUTH class, the installation has activated the SERVAUTH class, a profile covering this resource name has been created in the SERVAUTH class, and the effective user ID is permitted to the resource, then it is allowed to access the network.

Steps for modifying

To modify any values on the NETACCESS statement, use a VARY TCPIP,,OBEYFILE command with a data set that contains a new NETACCESS statement. All existing network entries are deleted and replaced with the entries from the new NETACCESS statement. Active connections are reauthorized whenever the user ID the active connections are running under has changed or a new NETACCESS statement is loaded.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands .

Statement dependency

- A security server must be running and the SERVAUTH class must be active or all users are denied access to all network addresses mapped to a security zone.
- A resource profile name must be defined for a security zone or all users are denied access to all network addresses mapped to that security zone.
- Each user must be authorized to the security zone containing their static or Dynamic IP address.
- Servers such as HTTPD, FTPD, and INETD must have the user ID they accept work under authorized to all security zones that contain their intended clients' addresses.
- The FTP anonymous user (ANONYMO) must be authorized to the security zones containing clients that are allowed anonymous access.
- Users must be authorized to the security zone containing the name server address they use to avoid resolver failures.

- To protect security zone definitions, authority to modify the initial profile data set and issue VARY TCPIP,,OBEYFILE commands must be controlled.
- When local addresses, or the DEFAULTHOME or DEFAULT parameters are specified and inbound checking is enabled, servers and other applications that explicitly bind must be permitted to the bind address.
  - Define address 127.0.0.1/8 or address ::1/128 into a security zone to control binds to the IPv4 or IPv6 loopback addresses, respectively.
  - Define address 0.0.0.0/32 or address ::/128 into a security zone to control binds to the IPv4 INADDR_ANY address, or to the IPv6 unspecified address (in6addr_any), respectively.
  - Use the BIND parameter on the PORT statement to optionally override binds to the IPv4 INADDR_ANY address, or to the IPv6 unspecified address (in6addr_any), with a bind to the specific local address specified on the BIND parameter. Permit the job to the security zone for that address.
- An IPv6 address should not be configured unless the TCP/IP stack is IPv6 enabled. If the stack is not IPv6 enabled, then all entries following an IPv6 entry are ignored and a message is issued.

Examples

```
NETACCESS    INBOUND    OUTBOUND  CACHEPERMIT ; check both ways, cache permits only
  192.168.0.0/16                    CORPNET ; Net address
  192.168.113.19/32                 HOST1   ; Specific host address
  192.168.113.0     255.255.255.0   SUBNET1 ; Subnet address
  192.168.112.0     255.255.248.0   SUBNET2 ; Subnet address
  192.168.192.0/24                  CAMPUS  ; Subnet address
  192.168.214.0/24                  CAMPUS  ; Subnet address
  fe80::6:2900:1dc:21bc/128         HOST2   ; IPv6 specific host address
  2001:0DB8::/16                     GLBL    ; IPv6 global network
  DEFAULTHOME                       HOME    ; Optional Default local zone
DEFAULT                             DEFZONE ; Optional Default zone
ENDNETACCESS
```

Usage notes

- The NETACCESS statement is optional.
- The initial profile or a VARY TCPIP,,OBEYFILE command data set can contain multiple NETACCESS statements.
- The first NETACCESS statement of each configuration data set that is executed resets the flags to OUTBOUND, NOINBOUND, and CACHEALL and clears any existing NETACCESS list prior to processing the flags and entries in that statement.
- Subsequent NETACCESS statements in the same configuration data set override any flags specified and add or replace specified entries in the list. Default flag values do not override previously specified values
- Specifying a DEFAULT is optional. If you do not specify a default, Network Access Control applies only to the networks which are explicitly listed in NETACCESS statements.
- When an incorrect NETACCESS entry is encountered, all entries following that entry in that NETACCESS statement are ignored. IPv4 entries as well as any DEFAULT and DEFAULTHOME entries should precede the first IPv6 entry, to ensure that they are accepted, if the TCP/IP stack is not IPv6 enabled.
- If the new NETACCESS list is empty at the end of the configuration data set, Network Access Control is disabled.

NETMONITOR statement

Use the NETMONITOR PROFILE.TCPIP statement to activate or deactivate selected real-time TCP/IP network management interfaces (NMI). See Real-time TCP/IP network monitoring NMI in z/OS Communications Server: IP Programmer's Guide and Reference for more information about these services.

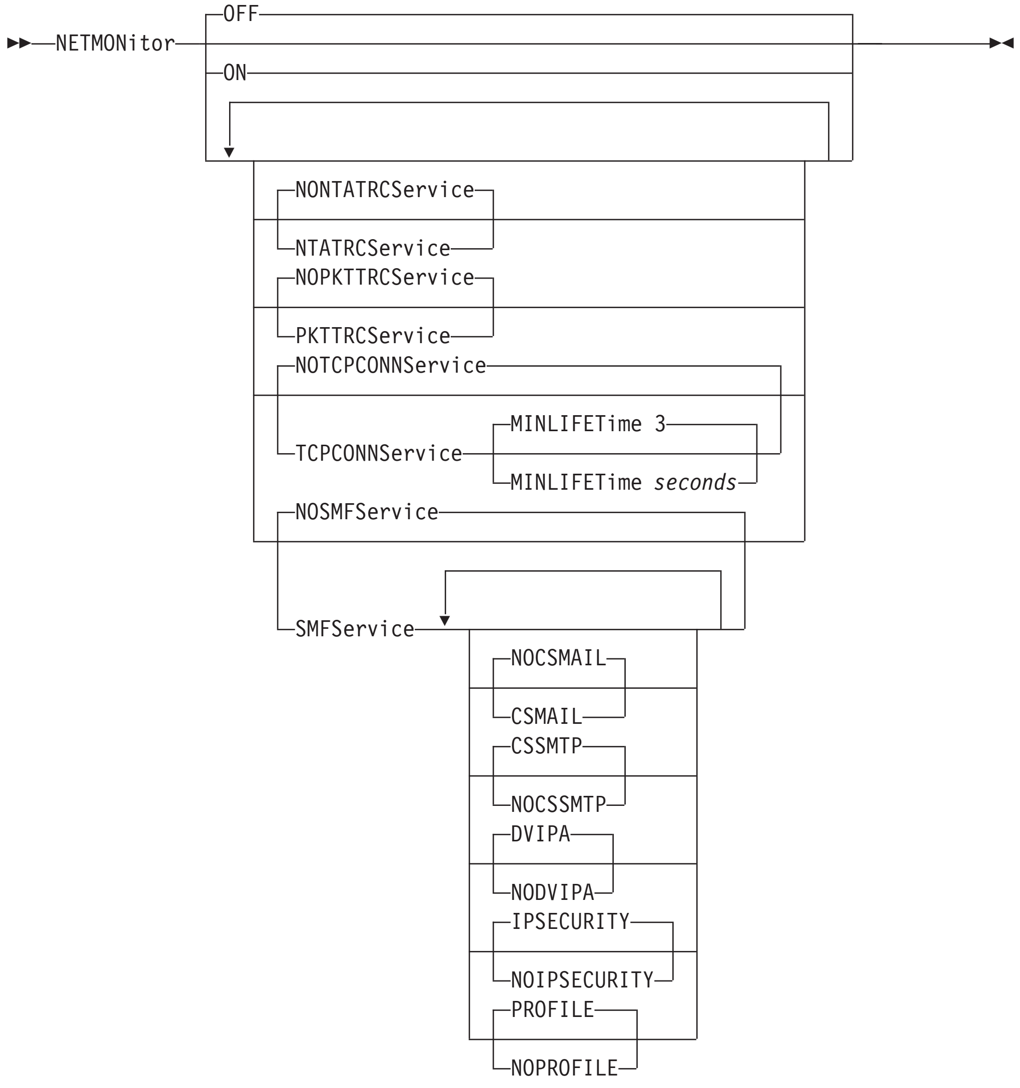
The NETMONITOR parameters, TCPCONNSERVICE and SMFSERVICE, provide two functions:

- They control the availability of the real time SMF services that are associated with each parameter.
- They control the creation of the SMF 119 records that are supported by each service.

If you want your application to process only SMF 119 records by using these real time SMF services, you need to configure only the NETMONITOR profile statement. You do not need to request support for these SMF 119 records on the SMFCONFIG profile statement.

Syntax

Tip: Specify the parameters for this statement in any order.



Parameters

OFF

If specified in PROFILE.TCPIP at initialization, indicates that no supported network monitoring services should be started. This is the default value.

If specified using the VARY TCPIP,,OBEYFILE command and any network monitoring services are currently enabled, then those services are disabled, causing all current client connections to those services to be terminated.

ON If specified in PROFILE.TCPIP at initialization, indicates that all supported network monitoring services should be started, and any options specific to those services (such as MINLIFETIME) set to their default values. If specified using the VARY TCPIP,,OBEYFILE command, then all network monitoring services that are not currently enabled are started, with any options specific to those services (such as MINLIFETIME) set to their default values. All services that are currently running at the time of a VARY TCPIP,,OBEYFILE command remain running, and any options specific to those services (such as MINLIFETIME) are unchanged. No other monitoring services are allowed on the NETMONITOR statement when the ON parameter is set.

NTATRCSERVICE | NONTATRCSERVICE
Specifies the behavior of the real time TCP/IP OSAENTA trace service (SYSTCPOT).

NONTATRCSERVICE
If this parameter is specified in PROFILE.TCPIP at initialization, it indicates that the OSAENTA trace service should not be activated on the stack. This is the default value. If specified using the VARY TCPIP,,OBEYFILE command and the OSAENTA service is currently enabled, the connections of the client applications are terminated and new connections are not accepted.

NTATRCSERVICE
Enables the OSAENTA trace service function to run on this TCP/IP stack. This service enables network management applications to access trace data that is collected for all OSAENTA traces. Access control should be provided for this service; see the z/OS Communications Server: IP Configuration Guide security topic for more information.

Tip: To ensure that a network management application that uses the real-time TCP/IP network management interface (NMI) receives OSAENTA data, verify the following requirements:

- The CTRACE SYSTCPOT component must be active. The component is activated by default. The TRACE CT,ON,COMP=SYSTCPOT,SUB=(tcpprocname) command activates the trace and the TRACE CT,OFF,COMP=SYSTCPOT,SUB=(tcpprocname) command deactivates the trace.
- The VARY TCPIP,,OSAENTA,ON command must be issued, or the OSAENTA statement in the TCP/IP profile is used to specify the parameters to collect data from an OSA-Express adapter. For trace data collection from an OSA-Express adapter, verify the following requirements:
  - An extra DATAPATH device must be defined for the TRLE for the OSA-Express adapter.
  - The hardware definitions for the OSA-Express adapter must be defined to allow OSAENTA trace data to be collected.

For more information, see Real-time TCP/IP network monitoring NMI in z/OS Communications Server: IP Programmer's Guide and Reference.

PKTTRCSERVICE | NOPKTTRCSERVICE
Specifies the behavior of the real time TCP/IP packet trace service (SYSTCPDA).

NOPKTTRCSERVICE
If specified in PROFILE.TCPIP at initialization, this parameter indicates that the packet trace service should not be allowed on the stack. This is the default value. If specified using the VARY TCPIP,,OBEYFILE command and packet trace service is currently enabled, the client applications connections are terminated and new connections are not accepted.

PKTTRCSERVICE
Enables the packet trace service function to run on this TCP/IP stack. This service enables network management applications to access trace data collected for any active packet traces or data traces. Access control

should be provided for this service; see the z/OS Communications Server: IP Configuration Guide security topic for more information.

Tip: To ensure that a network management application that uses the real-time TCP/IP NMI receives packet trace (PKTTRACE) or data trace (DATTRACE) data, verify the following requirements:

- The CTRACE SYSTCPDA component must be active. The component is activated by default when TCP/IP starts. The TRACE CT,ON,COMP=SYSTCPDA,SUB=(tcpprocname) command activates the trace and the TRACE CT,OFF,COMP=SYSTCPDA,SUB=(tcpprocname) command deactivates the trace.
- If the network management application collects packet trace data, the VARY TCPIP,,PKTTRACE,ON command must be issued to specify the parameters to collect trace data from TCP/IP interfaces.
- If the network management application collects data trace data, the VARY TCPIP,,DATTRACE,ON command must be issued to specify the parameters to collect application data.

For more information, see Real-time TCP/IP network monitoring NMI in z/OS Communications Server: IP Programmer's Guide and Reference.

TCPCONNSERVICE | NOTCPCONNSERVICE

Controls the behavior of the real time TCP connection SMF NMI service (SYSTCPCN), and the generation of SMF 119 records that are supported on this service.

NOTCPCONNSERVICE

If the parameter is specified in PROFILE.TCPIP at initialization, this parameter indicates that the TCP connection SMF NMI service should not be started on the stack. This is the default value. If the parameter is specified by the VARY TCPIP,,OBEYFILE command, this parameter indicates that the service should be stopped.

TCPCONNSERVICE

Enables the real time TCP connection SMF NMI service to run on this TCP/IP stack. The service runs as a subtask in the TCP/IP stack address space. This service provides an interface for network management applications to obtain information about TCP connections on this stack. For more information about this service and a list of the SMF 119 records supported on this service, see Real-time TCP/IP network monitoring NMI in z/OS Communications Server: IP Programmer's Guide and Reference.

Enabling or disabling this service has no effect on SMF 119 records written to the MVS SMF data sets due to the SMFCONFIG profile statement. You should provide access control for this service. For more information, see z/OS Communications Server: IP Configuration Guide.

MINLIFETIME *seconds*

The minimum connection lifetime, specified in seconds, for connections reported by the TCP connection information server. The server waits for this period before recording information about new connections; if the connection has closed in the meantime, then the connection is not reported by the TCP connection information server. This parameter can have a value from 0 to 60 seconds; the default, if not specified, is 3 seconds.

If 0 is coded for this option, then all connections are reported.

This option is used to suppress short-lived connections from being reported over the TCP connection information service. In order to ensure that all connections are reported, a 0 should be coded for this option.

SMFSERVICE | NOSMFSERVICE

Controls the behavior of the real time SMF NMI service (SYSTCPSM), and the generation of SMF 119 records that are supported on this service.

NOSMFSERVICE

If the parameter is specified in PROFILE.TCPIP at initialization, this parameter indicates that the real time SMF NMI service should not be started on the stack. This is the default value. If the parameter is specified using the VARY TCPIP,,OBEYFILE command, this parameter indicates that the service should be terminated.

SMFSERVICE

Enables the real time SMF NMI service to run on this TCP/IP stack. The service runs as a subtask in the TCP/IP stack address space. This service provides an interface for network management applications to obtain stack information in the form of SMF 119 records. This parameter can also be used, with or without subparameters, to request the creation of specific SMF 119 records which are then provided to applications that are connected to this service. For more information about this service and a list of all the SMF 119 records supported on this service, see Real-time TCP/IP network monitoring NMI in z/OS Communications Server: IP Programmer's Guide and Reference.

Certain FTP and Telnet SMF records are created by default. You must specify NOSMFSERVICE to stop the creation of these records. The creation of other SMF records is controlled by specifying the subparameter associated with the specific SMF record.

Rules:

- If you have specified the SMFSERVICE parameter (with or without subparameters), certain FTP and Telnet SMF records are created. There are no subparameters to control the creation of these SMF records.
- You can specify only the SMFSERVICE parameter, without any subparameters, to enable the creation of all supported SMF 119 records on this stack.
- For the SMF records whose creation is controlled by subparameters, specify the SMFSERVICE parameter with one or more subparameters to enable or disable the creation of the SMF records for the specified subparameter only.

Enabling or disabling this service has no effect on SMF 119 records written to the MVS SMF data sets due to the SMFCONFIG profile statement, or the FTP.DATA or CSSMTP SMF119 configuration statements. For more information, see z/OS Communications Server: IP Configuration Guide.

CSMAIL | NOCSMAIL

Controls the creation of the real time CSSMTP SMF MAIL records. (SMF 119 Subtype 50 (MAIL))

CSMAIL

Specifies that the real time CSSMTP SMF MAIL records should be created and provided on the real time SMF NMI service.

Special considerations for CSSMTP application and NETMONITOR in a CINET environment:

- The CSSMTP application can be started with the **-p** parameter to set stack name. All the SMF records are written to the real time SMF NMI associated with the stack whose name was specified on the **-p** parameter. This forces connection-oriented SMF records (CONNECT) and non-connection oriented SMF records (CONFIG, SPOOL, MAIL and STATS) to go to the same stack. The CSSMTP and CSMAIL NETMONITOR parameters should be specified in the profile data sets of the stack name whose name is specified on the **-p** parameter.
- If the CSSMTP application is started without the **-p** parameter and multiple stacks are active, then a network management application cannot determine the stack that records will be written to. So, the CSSMTP and CSMAIL NETMONITOR parameters should be specified in the profile data sets of all stacks, so that network management applications can obtain all the records. The network management application will not get redundant records.

NOCSMAIL

Specifies that the real time CSSMTP SMF MAIL records should not be created. This is the default.

CSSMTP | NOCSSMTP

Controls the creation of the real time CSSMTP SMF records. (SMF 119, Subtype 48 (CONFIG), 49 (CONNECT), 51 (SPOOL), and 52 (STATS)).

CSSMTP

Specifies that the real time CSSMTP SMF records should be created and provided on the real time SMF NMI service. This is the default.

Special considerations for CSSMTP application and NETMONITOR in a CINET environment:

- The CSSMTP application can be started with the **-p** parameter to set stack affinity. All the SMF records are written to the real time SMF NMI associated with the stack whose name was specified on the **-p** parameter. This forces connection-oriented SMF records (CONNECT) and non-connection oriented SMF records (CONFIG, SPOOL, MAIL and STATS) to go to the same stack. The CSSMTP and CSMAIL NETMONITOR parameters should be specified in the profile data sets of the stack name whose name is specified on the **-p** parameter.

- If the CSSMTP application is started without the **-p** parameter and multiple stacks are active, then a network management application cannot determine the stack that records will be written to. So, the CSSMTP and CSMAIL NETMONITOR parameters should be specified in the profile data sets of all stacks, so that network management applications can obtain all the records. The network management application will not get redundant records.

NOCSSMTP
Specifies that the real time CSSMTP SMF records should not be created.

DVIPA | NODVIPA
Controls the creation of the real time sysplex (dynamic virtual IP address) SMF records.

DVIPA
Specifies that the real time sysplex SMF records should be created and provided on the real time SMF NMI service.

NODVIPA
Specifies that the real time sysplex SMF records should not be created.

IPSECURITY | NOIPSECURITY
Controls the creation of the real-time IPSec SMF records.

IPSECURITY
Specifies that the real time IPSec SMF records should be created and provided on the real time SMF NMI service.

NOIPSECURITY
Specifies that the real time IPSec SMF records should not be created.

PROFILE | NOPROFILE
Controls the creation of the real-time TCP/IP stack profile SMF records and TN3270 Telnet server (Telnet) profile SMF records.

PROFILE
Specifies that the real-time TCP/IP stack profile SMF records (subtype 4) and TN3270 Telnet profile SMF records (subtype 24) should be created and provided on the real-time SMF NMI service.

NOPROFILE
Specifies that the real-time TCP/IP stack profile SMF records and TN3270 Telnet profile SMF records should not be created.

Steps for modifying

If NETMONITOR appears in a VARY TCPIP,,OBEYFILE command data set without any options, then no change occurs. NETMONITOR OFF must be explicitly coded in a VARY TCPIP,,OBEYFILE command data set in order to turn off all active services.

If a NETMONITOR statement in a VARY TCPIP,,OBEYFILE command data set contains service-specific keywords, then those services which are not specified on the statement remain unaffected by the command processing.

If any service is disabled by a VARY TCPIP,,OBEYFILE command, then clients connected to that service have their connections terminated.

If the MINLIFETIME setting is changed by a VARY TCPIP,,OBEYFILE command, then existing TCP connections are not affected by the new minimum lifetime value. Only new TCP connections are affected by the updated minimum lifetime value.

Related topic

- "SMFCONFIG statement" on page 274

OSAENTA statement

Use the OSAENTA statement to control the OSA-Express Network Traffic Analyzer (OEAENTA) tracing facility in the OSA-Express adapter. You can use this statement to select frames as candidates for tracing and subsequent analysis; OSAENTA traces are recorded externally using the TRACE command. See z/OS Communications Server: IP Diagnosis Guide for information about the steps required to perform an OSAENTA trace.

The OSAENTA statement consists of two parts. One part defines the OSA-Express that is to be traced and characteristics of the tracing. A second part turns tracing on or off or clears the trace settings. The tracing characteristics are identified by filters which specify under which conditions a frame should be traced. A frame must meet all the conditions specified on the OSAENTA statements for it to be traced. For example, if the OSAENTA statement identifies PROTOCOL=TCP and PORTNUM=21, only IP packets that have both a protocol of TCP and a port of 21 are traced. Only one value can be specified for a given filter on one OSAENTA statement.

Multiple OSAENTA statements can be included in the PROFILE.TCPIP data set, and can control tracing for multiple OSAs. The filters on multiple OSAENTA statements are cumulative for a given OSA-Express port. Each OSAENTA statement that specifies filters adds to the filters that are already in effect for that OSA-Express port. You can use multiple OSAENTA statements, multiple filter values can be assigned to each filter. There is a limit of eight filter values for each filter for each OSA-Express port. For example, you can specify up to eight IP protocols, up to eight VLAN IDs, and so on. For IP addresses, you can specify up to eight IPv4 addresses and up to eight IPv6 addresses. If a frame matches any of the values for that filter, it is considered to meet the condition of that particular filter. For example, if IPADDR=9.67.1.1,PROTO=TCP, and PORTNUM=21 is specified on one OSAENTA statement for OSA1, and IPADDR=9.67.1.2 is specified on another OSAENTA statement for OSA1, all frames sent to either IP address 9.67.1.1 or 9.67.1.2 with a protocol of TCP and a port of 21 are traced.

The OSAENTA statement dynamically defines a QDIO interface to the OSA-Express being traced, called an OSAENTA interface. That interface is used exclusively for capturing OSA-Express Network Traffic Analyzer traces.

The OSAENTA statement enables an installation to trace data from other hosts connected to OSA-Express. The trace data collected should be considered confidential and TCP/IP system dumps and external trace files containing this trace data should be protected.

If an error is found while parsing the OSAENTA statement, an error message is generated and the statement is ignored.

Syntax

Tip: Specify all parameters, except the PORTNAME parameter, for this statement in any order. If a keyword on a given statement is specified multiple times, the last value specified is used.

```
►►──OSAENTA──PORTNAME=osa_port_name──┬─────┬──┤ Parameter ├──┤ Filter ├──────────►◄
                                     ├─ON──┤
                                     ├─OFF─┤
                                     └─DEL─┘
```

Parameter:

```
                                                   ┌─DISCARD=EXCEPTION──┐
├──┬─────────────┬──┬────────────────────────────┬──┼────────────────────┼──────────►
   └─CLEARfilter─┘  └─DATA=──┬─1024─────────┬────┘  ├─DISCARD=ALL────────┤
                             └─trace_amount─┘       ├─DISCARD=NONE───────┤
                                                    └─DISCARD=discard_code┘

   ┌─FULL──────────────────────────┐  ┌─NOFILTER=NONE─┐
►──┼───────────────────────────────┼──┼───────────────┼──────────────────────────────►
   └─ABBREV=──┬─224───────────┬────┘  └─NOFILTER=ALL──┘
              └─abbrev_length─┘

►──┬──────────────────────────────┬──┬──────────────────────────┬──────────────────┤
   └─FRAMES=──┬─2147483647──┬─────┘  └─TIME=──┬─10080──────┬────┘
              └─trace_count─┘                 └─trace_time─┘
```

Filter:

```
   ┌─IPaddr=*─────────────────────────┐  ┌─DEVICEID=*─────────────┐
├──┼──────────────────────────────────┼──┼────────────────────────┼─────────────────►
   └─IPaddr=──┬─┤ IPv4_address ├─┬────┘  └─DEVICEID=device_id─────┘
              └─┤ IPv6_address ├─┘

   ┌─ETHType=*───────────────┐  ┌─MAC=*───────────┐  ┌─PORTNum=*───────────┐
►──┼─────────────────────────┼──┼─────────────────┼──┼─────────────────────┼─────────►
   ├─ETHType=IPV4────────────┤  └─MAC=mac_address─┘  └─PORTNum=port_number─┘
   ├─ETHType=IPV6────────────┤
   ├─ETHType=ARP─────────────┤
   ├─ETHType=SNA─────────────┤
   └─ETHType=ethernet_type───┘
   ┌─PROTOcol=*────────────────┐ ┌─VLANID=*───────┐
►──┼───────────────────────────┼─┼────────────────┼──────────────────────────────────┤
   ├─PROTOcol=TCP──────────────┤ ├─VLANID=vlan_id─┤
   ├─PROTOcol=UDP──────────────┤ └─VLANID=ALL─────┘
   ├─PROTOcol=ICMP─────────────┤
   ├─PROTOcol=ICMPv6───────────┤
   └─PROTOcol=protocol_number──┘
```

IPv4_address:

```
├──┬─ipv4_address─────────────────┬────────────────────────────────────────────────┤
   └─ipv4_address/num_mask_bits───┘
```

IPv6_address:

```
├──┬─ipv6_address─────────────────┬────────────────────────────────────────────────┤
   └─ipv6_address/prefix_length───┘
```

Parameters

PORTNAME=*osa_port_name*

Specifies the required port name of the OSA-Express for which you want to enable tracing. This is the same port name that is defined on the PORTNAME keyword of the VTAM TRLE statement. This is the same port name that is either defined on the PORTNAME keyword of the VTAM TRLE statement or is dynamically created by VTAM for OSX interfaces (configured with the CHPID parameter) or for OSM interfaces. For more information about OSM and OSX interfaces, see z/OS Communications Server: IP Configuration Guide..

Tip: You do not also have to define the OSA-Express to TCP/IP using the DEVICE or LINK statements or INTERFACE statement or activate it on the tracing stack in order to collect trace data from other stacks using that OSA-Express. For an OSX interface configured with the CHPID parameter or for an OSM interface, specify the port name according to the VTAM naming convention for these dynamic TRLEs, and VTAM will dynamically create the TRLE when you activate the OSAENTA interface. For details about the naming convention for these dynamically generated TRLEs, see z/OS Communications Server: SNA Network Implementation Guide.

Restriction: OSA-Express does not allow multiple stacks to concurrently use the tracing function for a given OSA-Express feature.

ABBREV

Specifies the amount of data that is to be traced for each frame. You can specify a length in the range 64 - 65472 or use the default value 224. The value is rounded up to the next 32 byte boundary. The ABBREV parameter can be used to control the volume of data stored in the trace buffers and file. The actual amount of data traced might be limited by the OSA adapter.

Tip: The size value that OSA returns is the maximum amount of trace data that OSA can return. Depending on the model, OSA can return fewer bytes than the maximum. OSA-Express3 or later version returns only 120 bytes for unicast packets, and return up to the maximum amount of trace data for multicast, broadcast, or unrouteable packets.

Guideline: Use a large value or specify the FULL parameter if you want to maximize the amount of data traced for each packet; TCP segmentation offload packets are traced before the packet is segmented, and can be larger than the largest frame size on the LAN. See the segmentation offload information in z/OS Communications Server: IP Configuration Guide for information about which parameters affect the size of TCP segmentation offload packets.

CLEARFILTER
Clears any previous OSAENTA trace filters for the specified OSA-Express port name.

If you specify the CLEARFILTER parameter and the OSAENTA interface is active, either all frames are traced or no frames are traced, depending on the setting of the NOFILTER parameter. The trace buffers are likely to fill up very quickly if you clear all the filters without setting new filters to filter out an adequate percentage of the packets.

Tip: Specifying CLEARFILTER clears all filters. To clear all values for a single filter, use OSAENTA and specify an asterisk (*) for the filter value.

DATA
Specifies the number of megabytes of data to be collected before stopping the trace. The minimum value is 1 megabyte, the default value is 1024 megabytes, and the maximum value is 2147483647 megabytes. If a value of 0 is specified, then the maximum value is set.

Result: If the OSAENTA interface is inactive, then the data limit takes effect when the OSAENTA trace is enabled with the ON parameter. If the OSAENTA interface is active and the *trace_amount* value is modified, then the stack resets the data counter to 0 and puts the new data limit into effect.

DEL
Removes the OSAENTA interface definition. The OSAENTA interface must be inactive in order to specify the DELETE parameter. To deactivate the OSAENTA interface, you can respecify the OSAENTA statement with the OFF parameter, or use the V TCPIP,,OSAENTA command with the OFF parameter.

DEVICEID
Specifies the 8-digit hexadecimal value that identifies a host that is sharing the OSA-Express feature. The value is in the form *csmfclus*, where:

cs	The channel subsystem ID for this datapath device.
mf	The LPAR Multiple Image Facility ID for the LPAR using this datapath device.
cl	The control unit logical identifier for this datapath device.
ua	The unit address for this datapath device.

Each identifier is a 2-digit hexadecimal value in the range 00 - FF.

If the frame was either inbound or outbound to the host identified by the *device_id* value, then the frame meets the criteria for this filter. If the DEVICEID option has been omitted or an asterisk (*) is specified, then all packets meet the criteria for this filter.

Tip: You can obtain the *device_id* values for any user of the OSA-Express feature by using the Hardware Management Console (HMC). For a data device that is active on a z/OS stack, you can obtain the *device_id* value for that data device from message IST2190I in the output from the D NET,TRL,TRLE=name command.

DISCARD

Specifies which frames that are discarded by the OSA-Express feature should be traced. Discarded frames include frames that OSA-Express feature could not transmit outbound or could not forward inbound. Discarded frames that match the DISCARD= setting are traced whether or not they match any filters that might be in effect. You can specify the DISCARD parameter on multiple OSAENTA statements. The ALL and NONE values reset any previous DISCARD values that are in effect, and the EXCEPTION value or a discard code resets the setting ALL or NONE. The EXCEPTION value and discard_code values are cumulative for a given OSA-Express feature.

ALL Specifies that all frames that are discarded by the OSA-Express feature are traced. This includes both exception conditions and expected discards, such as ARP packets received for non-registered IP addresses or packets for non-supported ethernet types.

EXCEPTION

Specifies that frames discarded by the OSA-Express feature for exception conditions are traced. These are frames that are typically discarded for anomalous conditions. Examples of anomalous conditions are:

- An inbound IP packet destined for an IP address that is not registered with the OSA-Express feature and no PRIROUTER or SECROUTER parameter is in effect.
- An outbound IP packet that could not be delivered because no storage was available within the OSA-Express feature.

Rule: If the EXCEPTION value and discard codes are specified on multiple OSAENTA statements, all frames that are discarded for exception conditions and all frames that are discarded for any of the discard codes in effect are traced.

Restriction: When the EXCEPTION value is specified, only seven or fewer discard codes can be active for one OSA-Express feature.

NONE

Specifies that no discarded frames are traced.

discard_code

Specifies that frames discarded for the reason specified by the *discard_code* value are traced. This option should be used only under the direction of IBM service personnel. Valid values are in the range 1 - 4087. As many as 8 discard codes can be active for one OSA-Express feature.

Rule: The CLEARFILTER parameter does not affect the state of the DISCARD parameter.

Result: A frame can be traced twice; once when the packet is passed to the OSA-Express feature, and again as a discarded packet during the processing of the packet.

Guideline: To reset the current set of active discard codes, specify DISCARD=ALL or NONE followed by OSAENTA statements with the DISCARD parameters that you want to specify.

ETHTYPE

Specifies the Ethernet frame type to be traced. This can be specified as one of the literals IPV4, IPV6, ARP, SNA or as a hexadecimal number in the range

0600 - FFFF (IPV4=0800, IPV6=86DD, ARP=0806, and SNA=80D5). If the ETHTYPE parameter has been omitted or an asterisk (*) is specified, then all packets meet the criteria for this filter.

FRAMES

Specifies the number of frames to be recorded before tracing is stopped. The minimum value is 100 frames. The maximum value is 2147483647 frames. If a value of 0 is specified, then the maximum value is set.

Result: If the OSAENTA interface is inactive, then the FRAMES limit takes effect when the OSAENTA trace is enabled with the ON parameter. If the OSAENTA interface is active and the *trace_count* value is modified, then the stack resets the frame counter to 0 and puts the new frame limit into effect.

FULL

Specifies that the entire frame is to be traced if possible. (OSA-Express might limit the amount of data actually traced.)

IPADDR

Specifies an IP address (either a 32-bit IPv4 address in dotted decimal notation, or a 128-bit IPv6 address in colon hexadecimal notation) to be compared with both the source and destination addresses of inbound and outbound packets. If either the source or destination address of a packet matches the specified IP address, the frame meets the criteria for this filter. If the IPADDR option is omitted or an asterisk (*) is specified, then all packets meet the criteria for this filter. If the IPADDR filter is specified, then only frames containing IP packets or ARP packets are subject to tracing.

If an IPv4 address is specified, then the /*num_mask_bits* variable (range 1-32) can be used to designate a subnet. The default number of bits is 32.

If an IPv6 address is specified, then an optional *prefix_length* value (range 1-128) can be specified. The default *prefix_length* value is 128.

Note:

1. If an IP address has never been specified on the OSAENTA command for the OSA portname, IPADDR=* is the default.
2. If IPADDR is specified on the command, you can specify one of the following values:
   - *
   - An IPv4 address
   - An IPv6 address

   There is no default if the IPADDR parameter is specified alone. If this parameter is specified by itself, it is a syntax error . Specify IPADDR=* to remove all previous IP addresses from the filter. Specify IPADDR=*IPv4_address* or IPADDRr=*IPv6_address* to add this address to the list of addresses for the IP address filter.

MAC

Specifies the twelve hexadecimal digits of the MAC address. The address is compared with both the source and destination MAC address of inbound and outbound frames. If either the source or destination address of a frame matches the specified MAC address, the frame meets the criteria for this filter. If the MAC option has been omitted or an asterisk (*) is specified, then all packets meet the criteria for this filter.

NOFILTER=ALL|NONE

Specifies the filtering behavior when all filters (DEVICEID, MAC, ETHTYPE, VLANID, IPADDR, PROTOCOL and PORTNUM) have been cleared or are inactive. This condition might exist if no filters have been specified, if the CLEARFILTER parameter is specified, or when the current setting for every filter is set to an asterisk (*). When NOFILTER=ALL, all packets are traced. When NOFILTER=NONE is specified, no packets are traced. The NOFILTER parameter applies only to packets that were not discarded by the OSA-Express feature. The DISCARD parameter controls tracing of discarded packets.

Guideline: If you clear filters using the CLEARFILTER parameter with the OSAENTA interface active, and specify NOFILTER=ALL, ensure that you also specify sufficient new filters. The trace buffers are likely to fill up very quickly if you clear all the filters without setting new filters to filter out an adequate percentage of the packets.

OFF
Disables OSA-Express feature tracing for the specified OSA-Express feature port name by stopping the OSAENTA interface. The trace parameters and filters remain in effect if you subsequently re-enable the OSAENTA trace.

ON Enables OSA-Express feature tracing for the specified OSA port name by starting the OSAENTA interface using the OSAENTA trace parameters and filters that are currently in effect. If the OSAENTA interface is already active, then the ON keyword causes the stack to reset the active counters on the DATA, FRAMES, and TIME limits.

Guideline: Ensure that you have specified sufficient trace filters before starting the trace. The trace buffers are likely to fill up very quickly if you activate the trace with either no filters (NOFILTER=ALL) or with a set of filters that does not filter out an adequate percentage of the packets.

PORTNUM
Specifies a port number in the range 1 - 65535. The port number is compared with the destination or source port of inbound and outbound packets. If the port of a packet is the same as the specified port number, then the frame meets the criteria for this filter. This comparison is performed only for packets using the TCP or UDP protocol; frames using other protocols are not traced when a PORTNum filter is in effect. If the PORTNum parameter is omitted or an asterisk (*) has been specified, then all packets meet the criteria for this filter. If the PORTNum filter is used, then only frames containing IP packets are subject to tracing.

IPSec Encapsulating Security Payload (ESP) packets cannot be traced by port number because the TCP or UDP headers are encrypted.

PROTOCOL
Specifies the IP protocol type to be traced. This can be specified as one of the literals TCP, UDP, ICMP, or ICMPV6, or as a number in the range 0 - 255 (ICMP=1, TCP=6, UDP=17, ICMPV6=58). If the PROTOCOL parameter is omitted or an asterisk (*) has been specified, then all packets meet the criteria for this filter. If a PROTOCOL protocol value is specified and the frame does not contain an IP protocol packet, then the frame is not traced. If the PROTOCOL filter is used, then only frames containing IP packets are subject to tracing.

Rule: For encapsulated packets, OSAENTA bases collection on whether the specified protocol filter matches the outermost packet protocol. For example, if TCP was specified as the protocol filter, and a TCP packet was received encapsulated in an IPSEC packet with protocol 50, this TCP packet is not collected. Protocol 50 must be specified to collect these packets.

TIME

Specifies the number of minutes that trace records are recorded before stopping. The minimum value is 1 minute. The maximum value is 10080 minutes (7 days). If a value of 0 is specified, then the maximum value is set.

Result: If the OSAENTA interface is inactive, then the time limit takes effect when the OSAENTA trace is enabled with the ON parameter. If the OSAENTA interface is active and the *trace_time* value is modified, then the stack resets the time counter to 0 and puts the new time limit into effect.

VLANID

Specifies a VLAN identifier value, which is a decimal number in the range 0 - 4094. The keyword ALL indicates that all frames that have a VLAN tag are included. If the VLANID parameter has been omitted or an asterisk(*) is specified, then all frames meet the criteria for this filter. If a VLAN identifier is specified, and the frame does not contain a VLAN tag or does not match the VLAN identifier, then the frame is not traced.

Steps for modifying

As previously indicated, the OSAENTA statements are cumulative for a given OSA-Express adapter, and any subsequent OSAENTA statement processed adds to the filters that are already in effect for that OSA-Express feature. To actually change a value for a given filter, the following options are available:

- Define an OSAENTA statement with a filter value of *, effectively deleting all values for that one filter entirely. Then define subsequent OSAENTA statements with the new filter values.
- Define an OSAENTA statement with the CLEARFILTER parameter, which removes all existing filters, and specify the entire list of filter attributes.

Tip: If the trace is currently enabled, the trace continues to run while each filter is modified or added. This can become an issue when changing a value for a given filter. Because both options involve deleting current filters, more data than you want is being traced during this time. Turn the trace off (define an OSAENTA statement with the OFF option) before changing filter values.

Examples for enabling, disabling, and modifying the OSA-Express feature tracing facility are shown in "Examples" on page 249.

You can also modify existing OSAENTA settings by using the VARY TCPIP,,OSAENTA command. See z/OS Communications Server: IP System Administrator's Commands for more information. Use the Netstat DEvlinks/**-d** command to display the results.

Usage notes

- You can use the Netstat DEvlinks/**-d** command to display the current OSAENTA trace settings.
- When the DATA, FRAMES, or TIME values are exceeded, the stack disables the OSAENTA trace, but this does not happen immediately. Trace records from the OSA-Express feature continue to be recorded until the stack has successfully contacted the adapter to stop the OSAENTA trace.
- To verify that the Ctrace component SYSTCPOT is active for a stack, issue DISPLAY TRACE,COMP=SYSTCPOT,SUB=(*tcpip_procname*).

- To write the data to the external writer, use the MVS TRACE,CT,WTRSTART=*writer_procedure* command to start the writer and the TRACE CT,ON,COMP=SYSTCPOT,SUB=(*tcpip_procname*) command to connect to the writer.
- The last buffer trace data are not written to the external writer until the writer has been disconnected from TCPIP and stopped.
- The TRACE CT,OFF,COMP=SYSTCPOT,SUB=(*tcpip_procname*) command stops the recording of trace data into TCPIP buffers and to the external writer. It does not stop the receipt of trace data from the OSA-Express feature. Issue a TRACE ON command to start recording the trace data into the buffers. To halt the receipt of trace data from the OSA-Express feature, specify the OSAENTA statement with the OFF parameter, or use the V TCPIP,,OSAENTA command with the OFF parameter.
- The OSAENTA trace can have performance implications if sufficient trace filters are not specified before enabling the trace. OSAENTA can reduce the amount of traffic the OSA-Express device can process and the amount of traffic that can be accelerated through that OSA-Express device. Also, host processing to collect the OSAENTA trace records can increase host CPU consumption. Specify sufficient filters to limit the amount of traffic traced to what is necessary for problem diagnosis.

The following differences exist between the OSAENTA and PKTTRACE statements:
- The PKTTRACE statement can collect data for only a single TCP/IP stack. The OSAENTA statement can collect data for other stacks that share the OSA-Express feature.
- Data collection enabled with the PKTTRAACE statement starts immediately. The data collection enabled with the OSAENTA statement is not started until the ON parameter is used.
- Each PKTTRACE command or statement is one set of filters. OSAENTA statement filters accumulate across multiple OSAENTA commands or statements.

Examples

The following sample includes several examples of the OSAENTA statement:

```
;
     ;
     ; set up the filters to trace for TCP packets on PORT 5003 with a source
     ;or destination
     ; IP address of 9.67.116.124 over MAC address 000084576893
     OSAENTA PORTNAME=OSA4 PROT=TCP IP=9.67.116.124 PORTNUM=5003
     OSAENTA PORTNAME=OSA4 MAC=000084576893
     ; activate the tracing (the trace will self-deactivate after 20,000 frames)
     OSAENTA PORTNAME=OSA4 ON FRAMES=20000
     ;
     ; deactivate the tracing
     OSAENTA OFF PORTNAME=OSA4
     ;
     ; Reactivate the tracing for another 20,000 frames
     OSAENTA ON PORTNAME=OSA4
     ;
     ; Modify tracing to change a port filter
     OSAENTA PORTNAME=OSA4 PORTNUM=*
     OSAENTA PORTNAME=OSA4 PORTNUM=21
     ;
     ; Change the parameters (add an IP address)
     OSAENTA IP=9.67.116.125 PORTNAME=OSA4
     ;
     ; Set up tracing for a new problem on OSA5
     ;  trace frames on VLAN 192 or 193 with an IPaddress 9.37.124.00 to .255 or
     ;                                                    9.37.125.00 to .255 or
     ;                                                    9.37.126.00 to .255
     OSAENTA PORTNAME=OSA5 ABBREV=480 TIME=5
           VLANID=192 IP=9.37.124/24
     OSAENTA PORTNAME=OSA5          IP=9.37.125/24
     OSAENTA PORTNAME=OSA5 VLANID=193 IP=9.37.126/24

     ; Now activate the trace with the new filters for 5 minutes
     OSAENTA ON PORTNAME=OSA5

 ; Reset the VLANID filter and restart tracing for another 5 minute interval
 OSAENTA ON PORTNAME=OSA5 VLANID=*
```

Figure 2. Example of the OSAENTA statement

Related topics

- V TCPIP,,OSAENTA command in z/OS Communications Server: IP System Administrator's Commands
- OSAENTA Trace in z/OS Communications Server: IP Diagnosis Guide
- OSA-Express Network Traffic Analyzer trace in z/OS Communications Server: IP Configuration Guide
- Display TRL command in z/OS Communications Server: SNA Operation

PKTTRACE statement

Use the PKTTRACE statement to control the packet tracing facility in TCP/IP. You can use this statement to select IP packets as candidates for tracing and subsequent analysis.

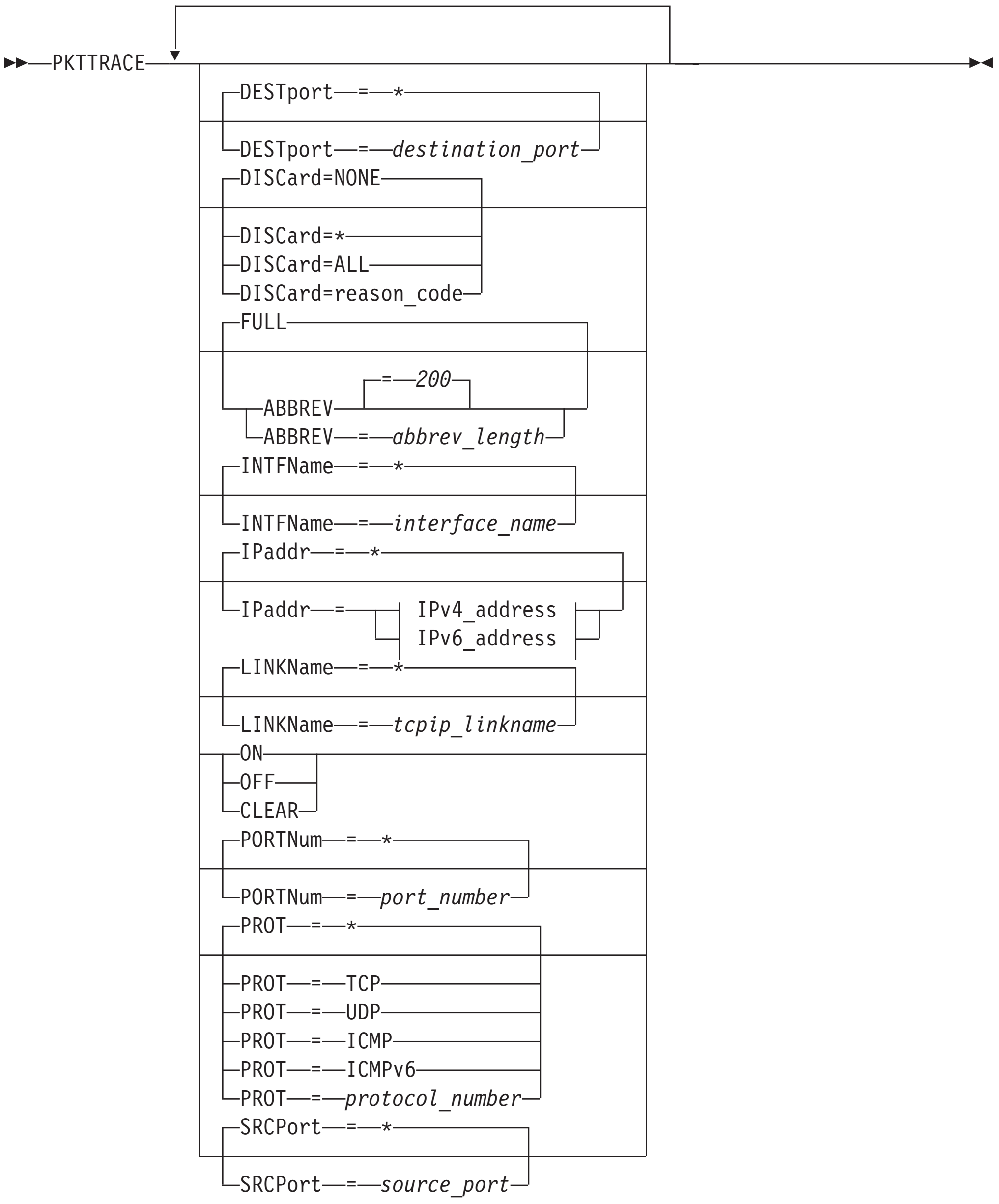
Restriction: An IP packet must meet all the conditions specified on the statement for it to be traced.

The PKTTRACE statement consists of two parts. The first part defines to TCP/IP the network interfaces that are to be traced and characteristics of how they are to be traced. The second part turns packet tracing ON or OFF or CLEARs packet trace settings for the interfaces specified on prior PKTTRACE statements or for a single interface if the LINKName/INTFName parameter is used.

Packet traces are recorded externally using the TRACE command CTRACE writer instead of GTF. See z/OS Communications Server: IP Diagnosis Guide for information about the steps required to perform an IP packet trace.

Syntax

Tip: Specify the parameters for this statement in any order.



IPv4_address:

```
                    ┌SUBNet=255.255.255.255┐
├──┬ipv4_address────┴──────────────────────┴─┬──────────────────────────────────────────┤
   ├ipv4_address──_SUBNet=──subnet_mask──────┤
   └ipv4_address/num_mask_bits───────────────┘
```

IPv6_address:

```
├──┬ipv6_address──────────────┬────────────────────────────────────────────────────────┤
   └ipv6_address/prefixLength─┘
```

Parameters

ABBREV
Specifies that a truncated portion of the IP packet is to be traced. You can specify a length in the range 0 - 65 535, or use the default of 200. The ABBREV parameter can be used to reduce the volume of data stored in the trace file.

The protocol headers are always included, even if they exceed the ABBREV value.

CLEAR
Disables packet tracing for the interfaces specified and removes the characteristics defining how they should be traced.

DESTPORT
Specifies a port number that is compared with the destination port of inbound and outbound packets. The port number is an integer in the range 1 - 65 535. If the destination port of a packet is the same as the specified port number, the packet is traced. This comparison is performed only for packets using the TCP or UDP protocol; packets using other protocols are not traced. If the DESTPORT parameter is omitted, and the PORTNUM parameter is also omitted, or an asterisk (*) is specified for the DESTPORT parameter, the destination port of packets is not checked.

IPSec Encapsulating Security Payload (ESP) packets cannot be traced by using the port number because the TCP or UDP headers are encrypted.

DISCARD
Specifies the IP packet discard reason code for the packets that should be traced. All IP packets have a discard reason code associated with them, which is typically set to 0. When the TCP/IP stack discards a packet, a specific discard reason code is set in this field. See the IP discard reason codes information in z/OS Communications Server: IP and SNA Codes for a list of all the discard reason codes. Typically, the TCP/IP stack does not trace discarded packets. You must specify a DISCARD value other than NONE to trace discarded packets. Valid values for DISCARD are:

*****
The DISCARD parameter is not applied to the selection of packets. All packets are traced.

ALL
Specifies that IP packets with a nonzero discard reason code should be traced. Specifying this value results in tracing only discarded packets.

NONE
Specifies that only IP packets that were not discarded should be traced. This is the default value.

reason_code
Specifies that only IP packets with the specified discard *reason_code*

value should be traced. The *reason_code* value is a number in the range of 4 096 - 20 479. You can also specify a value of 0, which is the equivalent of DISCARD=NONE.

Tips:

- A packet can be traced twice, once at the lower level IP layer when a packet arrives (with a discard reason code of 0), and again as a discarded packet in an upper level protocol layer of TCP/IP.
- You can use one packet trace profile statement per discard reason code. You can also specify a packet trace statement with DISCARD=ALL to trace all packets that are discarded. The other specified parameters are used to further select which discarded packets are traced. For example, use the following code to collect packets with discard reason code 4138 on all TCP or UDP packets with PORT number 20:

  ```
  PKTTRACE ON,DISCARD=4138,PORTNUM=20
  ```
- Specifying the SRCPORT, DESTPORT, IPADDR, PORTNUM or PROTOCOL parameters might prevent malformed packets from being traced.

FULL
Specifies that the entire IPADDR packet is to be traced.

IPADDR
Specifies an IPv4 or IPv6 address that is compared with both the source and destination addresses of inbound and outbound packets. If either the source or destination address of a packet matches the specified IP address, the packet is traced. If the IPADDR option is omitted, or an asterisk (*) is specified, then all IP addresses are traced.

Guidelines:

- If an IPv6 address is specified, an optional prefix length in the range 1 - 128 is allowed. The default prefix length is 128.
- If an IPv4 address is specified, the /num_mask_bits value is allowed.

/num_mask_bits
Specifies a numeric mask in the range 1 - 32.

/prefixLength
Specifies a numeric prefix length in the range 1 - 128.

LINKNAME|INTFNAME
The LINKNAME and INTFNAME parameters are interchangeable. They specify the name of the network interface defined on a preceding LINK or INTERFACE statement. If the LINKNAME or INTFNAME parameter is omitted or an asterisk (*) is specified for either parameter, the PKTTRACE parameters apply to all IPv4 and IPv6 interfaces prior to this statement.

To facilitate defining packet tracing when many interfaces are involved, use the PKTTRACE statement with the LINKNAME=* or INTFNAME=* option to define packet tracing characteristics for the majority of the interfaces. Then use individual PKTTRACE statements with specific LINKNAME or INTFNAME parameters for each interface that must be defined differently from the majority or interfaces.

The PKTTRACE statement must appear after a valid LINK or INTERFACE statement for the link or interface in the PROFILE.TCPIP data set.

OFF
Disables packet tracing for the specified interfaces and removes the characteristics defining how they should be traced.

If LINKNAME=* or INTFNAME=* and all other parameters are defaults, all trace structures are deactivated and removed from all existing IPv4 and IPv6 interfaces.

If LINKNAME=* or INTFNAME=* and PROT=UDP, all trace structures for all resources are analyzed; any matches are removed. If no trace structures remain, trace is deactivated for that resource.

If LINKNAME=*link_name* or INTFNAME=interface_name and there are no other parameters, all trace structures for *link_name/interface_name* are deactivated and removed.

If LINKNAME=*link_name* and IP=127.0.0.1, or INTFNAME=interface_name and IP=::1, then that particular trace structure is removed if it is found. If there is only one trace structure, then that structure is removed and trace is deactivated for that resource.

ON Turns on packet tracing, clears all settings previously defined and refreshes just the default settings.

If you use LINKNAME=* or INTFNAME=* and all other parameters are defaults, even if the defaults are specified, the command results replace any existing trace structures for all existing IPv4 and IPv6 interfaces.

If you use LINKNAME=*link_name* or INTFNAME=interface_name and another nondefault parameter, the command results are added to any existing trace structures. However, if the existing trace structure for *link_name/interface_name* is all defaults, the existing trace structures are discarded.

PORTNUM
Specifies a port number that is compared with the destination and source port of inbound and outbound packets. You can use this parameter instead of using the SRCPORT and DESTPORT parameters. The port number is an integer in the range 1 - 65 535. If the destination or source port of a packet is the same as the specified port number, the packet is traced. This comparison is performed only for packets using the TCP or UDP protocol; packets using other protocols are not traced. If the PORTNUM parameter is omitted and the SRCPORT and DESTPORT parameters are also omitted, then the port numbers of packets are not checked. If an asterisk (*) is specified, packets of any protocol and any destination or source port number are traced.

Guideline: SRCPORT and DESTPORT parameters should not be specified on the same PKTTRACE statement as the PORTNUM parameter. When the PORTNUM parameter is specified after DESTPORT or SRCPORT parameters, the DESTPORT and SRCPORT parameters are ignored.

Restriction: IPSec Encapsulating Security Payload (ESP) packets cannot be traced by port number because the TCP or UDP headers are encrypted.

PROT
Specifies the protocol type to be traced. This can be specified as one of the literals TCP, UDP, ICMP, or ICMPV6, or as a number between 1 and 255 (ICMP=1, TCP=6, UDP=17, ICMPV6=58, and RAW=255). If the PROT parameter is omitted or an asterisk (*) is specified, packets of any protocol are traced.

SRCPORT
Specifies a port number that is compared with the source port of inbound and outbound packets. The port number is an integer in the range 1 - 65535. If the source port of a packet is the same as the specified port number, the packet is traced. This comparison is performed only for packets using the TCP or UDP

protocol; packets using other protocols are not traced. If the SRCPORT parameter is omitted, and the PORTNUM parameter is also omitted, or an asterisk (*) is specified for the SRCPORT parameter, the source port of packets is not checked.

IPSec Encapsulating Security Payload (ESP) packets cannot be traced by port number because the TCP or UDP headers are encrypted.

SUBNET
Specifies a subnet mask that applies to the host and network portions of the IP address specified on the accompanying IPADDR parameter. The subnet mask must be specified in dotted decimal notation and must be specified in conjunction with the IPADDR parameter. The default is 255.255.255.255.

Steps for modifying

You can activate tracing at any time by executing the VARY TCPIP,,OBEYFILE command with a data set that contains PKTTRACE statements. However, the interface names specified on the PKTTRACE statements must already be defined. For example:

```
PKTTRACE ON,LINKNAME=*
LINK ...
DEVICE ...
```

In this example, the trace is done only for the LOOPBACK interface.

For more information about changing PKTTRACE parameters, see the descriptions for the ON and OFF parameters for “PKTTRACE statement” on page 250.

You can also modify existing PKTTRACE settings by using the VARY TCPIP,,PKTTRACE command. See z/OS Communications Server: IP System Administrator's Commands for more information.

To trace all the packets for a particular application port, enter two PKTTRACE commands:

```
PKTTRACE ON,DESTport=21
PKTTRACE ON,SRCport=21
```

The two commands capture all the packets received and all the packets sent for a particular port. If other options are specified, then they should be the same on both commands.

Use the Netstat DEvlinks/**-d** command to display the results. An IP packet is traced according to the first trace structure that the packet matches.

Statement dependency

- INTFName and LINKName are mutually exclusive. An error message is displayed if both are coded.
- The num_mask_bits and SUBNET= are mutually exclusive. An error message is displayed if both are coded.
- IP=* implies IP=0.0.0.0 and SUBNET=255.255.255.255.
- The IP address and subnet mask pair specified must be in the same network.
- Tracing is not done for packets whose destination and source IP address match. However, tracing is always done for packets using a loopback interface.

Usage notes

- Multiple PKTTRACE statements can be included in the PROFILE.TCPIP; the results are cumulative.
- If a keyword on a given statement is specified multiple times, the last value specified is used. If an option appears more than once on a statement, the value associated with the last occurrence of the option is used.
- If you do not specify any options on the PKTTRACE statement, all packets through all devices are traced except for discarded packets. The default is DISCARD=NONE.
- If an error is found while parsing the PKTTRACE statement, an error message is generated, the parameter in error is ignored, and the rest of the statement is parsed. If an error is produced by an incorrect ABBREV value, the ABBREV value is changed to the default.
- Each defined interface has an associated trace profile. The trace profile stores the values of each of the trace options for the interface. When you create or reset a trace profile for an interface using the CLEAR option, the trace profile is set to the default values for the trace options as follows:

  PROT
  All protocols

  IPADDDR
  All IP addresses

  SUBNET
  No checking

  SRCPORT
  No checking

  DESTPORT
  No checking

  FULL
  Trace of the whole IP packet

Examples

The following sample includes several examples of the PKTTRACE statement:

```
; CTC Device and Link
DEVICE CTC1     CTC     D00
LINK   CTCD00   CTC  1  CTC1
;
; CTC Device and Link
DEVICE CTC2     CTC     D02
LINK   CTCD02   CTC  1  CTC2
;
; CTC Device and Link
DEVICE CTC3     CTC     D04
LINK   CTCD04   CTC  1  CTC3
;
; LCS Device and Links
DEVICE LCS1     LCS     100
LINK   TR1      IBMTR        1  LCS1
LINK   LCSC00   ETHERNET     2  LCS1
LINK   LCSF00   FDDI         3  LCS1
;
DEVICE LCS2     LCS     102
LINK   LCS802   802.3        1  LCS2
;
DEVICE LCS3     LCS     104
```

```
LINK   LCSE802  ETHEROR802.3 1  LCS3
;
; start pkttrace
PKTTRACE ON LINKNAME=*
;
; set defaults for all links not specified below
PKTTRACE
; set for CTCD00
PKTTRACE FULL LINKNAME=CTCD00 PROT=* IP=* SRCPORT=* DESTPORT=*
; set for CTCD02
PKTTRACE ABBREV LINKNAME=CTCD02 PROT=TCP IP=9.67.116.124
              SRCPORT=5000 DESTPORT=161
; set for CTCD04
PKTTRACE ABBREV=1 LINKNAME=CTCD04 PROT=UDP IP=9.67.116.124
              SUBNET=255.255.255.255 SRCPORT=161 DESTPORT=5000
; set for TR1
PKTTRACE ABBREV=200 LINKNAME=TR1 PROT=ICMP IP=*
              SRCPORT=5000 DESTPORT=161
; set for LCSC00
PKTTRACE ABBREV=65535 LINKNAME=LCSC00 PROT=1 IP=9.67.116.124
              SUBNET=255.255.255.255 SRCPORT=* DESTPORT=*
; set for LCSF00 not to trace
PKTTRACE OFF LINKNAME=LCSF00
```

Related topics

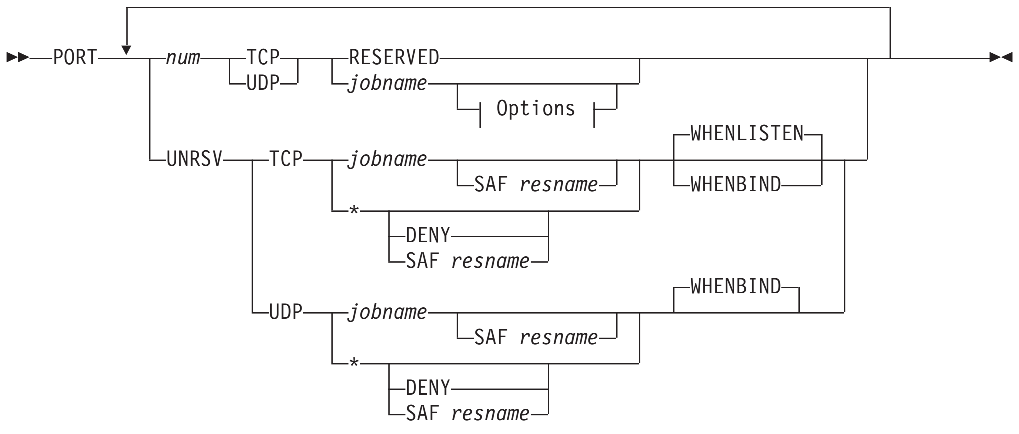
- "Summary of DEVICE and LINK statements" on page 47
- "Summary of INTERFACE statements" on page 141
- z/OS Communications Server: IP Diagnosis Guide

PORT statement

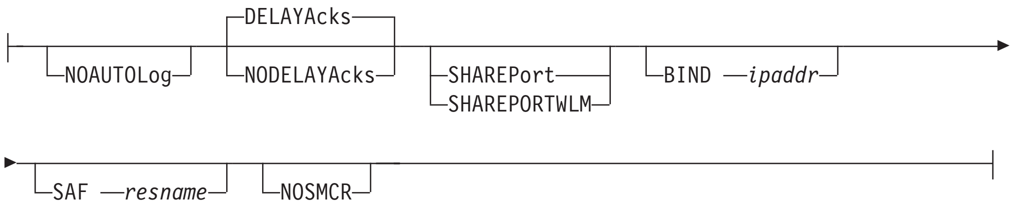
Use the PORT statement to reserve a port for one or more specified job names or to control application access to unreserved ports.

Syntax

Rule: The PORT parameters and options (for example, NOAUTOLOG, DELAYACKS) must be specified in the order in which they appear on the following syntax diagram.



Options:



Parameters

num

The number of the port to be reserved. The same port number can appear in more than one PORT statement with different users or more than once in the same PORT statement. This port cannot appear in a range specified by the PORTRANGE statement. If a PORTRANGE statement including this port number is specified prior to this statement, this port is ignored. If the PORTRANGE statement follows this statement, the PORTRANGE statement is ignored. An error message is generated in either case. *num* is a value in the range 1 - 65535.

Requirement: For z/OS UNIX applications that are invoked by INETD, ensure that the port number defined for the application in the /etc/services file is the same as the port number reserved for the application on the PORT statement.

UNRSV

This value indicates any unreserved port (any port number that is in the range 1 - 65535 that has not been reserved by a PORT or PORTRANGE statement).

Use PORT UNRSV statements to indicate which applications or users are permitted to access application-specified unreserved ports. PORT UNRSV statements control access to all unreserved ports in the range 1 - 65535 unless RESTRICTLOWPORTS is configured; however, when RESTRICTLOWPORTS is configured, PORT UNRSV statements control access to unreserved ports only above port 1023. For UDP, access control is applied when an application issues a bind to a particular port number to establish a local port. For TCP, access control is applied depending on the value of the WHENBIND or WHENLISTEN parameter.

If neither DENY nor the SAF keyword is specified, an application that matches the protocol and specified job name [the job name can be an asterisk (*)] on a PORT UNRSV statement can access unreserved ports. If DENY is specified, all applications are denied access to unreserved ports for the specified protocol. If the SAF keyword is specified, applications that match the PORT UNRSV statement must also have user access to the SAF SERVAUTH resource which is indicated by the SAF keyword, to be permitted to access an unreserved port.

Results:

- When no PORT UNRSV statements are configured for the socket protocol that is being used, all applications are allowed access to the unreserved ports unless prevented by TCPCONFIG or UDPCONFIG RESTRICTLOWPORTS or by GLOBALCONFIG EXPLICITBINDPORTRANGE. This is the default.
- When TCPCONFIG or UDPCONFIG RESTRICTLOWPORTS is configured for the access protocol that is being used, PORT UNRSV access control applies only to unreserved ports above port 1023.

- If any PORT UNRSV statements are configured for a protocol, access is determined by the PORT UNRSV statement whose specified job name most closely matches the application's job name. If the application's *jobname* does not match any of the PORT UNRSV statements, the application's access to unreserved ports is denied for that protocol.
- PORT UNRSV statements control access to nonzero, unreserved ports that are specified on explicit binds. Access to unreserved ports that are assigned by the stack is not affected.

Guideline: In a Common INET (CINET) environment with multiple stacks and no established stack affinity, an explicit bind to port 0 is converted to a bind to a specific port in the CINET range. If you have not reserved the ports in your CINET range for *jobname* OMVS, the explicit bind to port 0 is treated as an explicit bind to an unreserved port.

RESERVED
Indicates the port is not available for use by any user. Use RESERVED to lock certain ports. This is optional and valid for TCP or UDP protocols.

jobname
Specifies the MVS job name that can use the specified port (or any unreserved port in the case of a PORT UNRSV statement). You can specify the *jobname* value using a wildcard value consisting of 0 - 7 characters followed by an asterisk (*). For UDP, only one job name can be associated with a particular port. For TCP, the same port can be reserved multiple times for different job names. This can be useful if you have different servers with different job names that need access to the same port. For PORT UNRSV statements, both TCP and UDP can have multiple statements with different job names.

For multiple TCP reservations for the same port, or for multiple PORT UNRSV statements for the same protocol, the TCP/IP stack searches these PORT statements for the closest match (if any) to the application's job name. If you specified the job name using a wildcard on more than one of these statements, the TCP/IP stack matches the application job name to a PORT statement *jobname* value using the most specific value first and the least specific value (or value *, if it was specified) last.

Restriction: To reserve a port that is to be monitored by AUTOLOG, the *jobname* name must exactly match (no wildcards) the *jobname* name on the AUTOLOG statement.

The environment in which the application is run determines the job name to be associated with a particular client or server application.

The following list explains how to determine the *jobname* value given the environment in which the application is run:
- Applications run from batch use the batch job name.
- Applications started from the MVS operator console use the started procedure name as the job name.
- Applications run from a TSO user ID use the TSO user ID as the job name.
- Applications run from the z/OS shell normally have a job name that is the logged on user ID plus a one-character suffix.
- Authorized users can run applications from the z/OS shell and use the _BPX_JOBNAME environment variable to set the job name. In this case, the value specified for the environment variable is the job name.
- Use the name of the started JCL procedure for the UNIX System Services kernel address space to enable applications (except for applications using the

Pascal API) to bind to the port. This name is typically OMVS unless a different name is explicitly specified in the STARTUP_PROC parameter of the BPXPRMxx parmlib member.

- z/OS UNIX applications started by INETD use the *jobname* of the INETD server.
- Use the name of the VTAM started task for the UDP ports that are to be used for Enterprise Extender (EE) network connections.

  Restriction: The VTAM job name cannot include a wildcard character (*) when it reserves EE UDP ports.

Reserved Port Options:

NOAUTOLOG

Tells the TCP/IP address space *not* to restart the server if it was stopped previously. Otherwise, the default is to restart the server if it was stopped previously. If the application associated with the job name is an AUTOLOG started procedure, and the port is inactive (for TCP connections, the procedure must have a socket open to that port in the LISTEN state; for UDP connections, the procedure must have a socket open to that port), then AUTOLOG assumes that the procedure is hung; it cancels and restarts it every five minutes. Use NOAUTOLOG to prevent this from occurring. See "AUTOLOG statement" on page 23 for more information.

DELAYACKS | NODELAYACKS

DELAYACKS

Delays transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. The DELAYACKS parameter on the PORT statement affects only connections that use this port. This is the default, but the behavior can be overridden by specifying the NODELAYACKS parameter on the TCP/IP stack TCPCONFIG profile statement, or on any of the following statements used to configure the route used by the TCP connection:

- The TCP/IP stack BEGINROUTES or GATEWAY profile statements
- The Policy Agent RouteTable statement
- The OMPROUTE configuration statements

NODELAYACKS

Specifies that an acknowledgment is returned immediately when a packet is received with the PUSH bit on in the TCP header. The NODELAYACKS parameter on the PORT statement affects only connections that use this port. Specifying NODELAYACKS on the PORT statement overrides the specification of the DELAYACKS parameter on the TCP/IP stack TCPCONFIG profile statement or on any of the following statements used to configure the route used by the TCP connection:

- The TCP/IP stack BEGINROUTES or GATEWAY profile statements
- The Policy Agent RouteTable statement
- The OMPROUTE configuration statements

SHAREPORT

Required when reserving a port to be shared across multiple listeners on the same interface. When SHAREPORT is specified, TCP/IP allows multiple listeners to listen on the same combination of port and IP address.

As incoming client connections arrive for this port and IP address, TCP/IP distributes them across the listeners. Specification of this keyword causes incoming connection requests for the port to be distributed among the listeners using a weighted round-robin distribution method based on the servers' accept Efficiency Fractions (SEFs) of the listeners sharing the port. The SEF is a measure, calculated at intervals of approximately one minute, of the efficiency of the server application in accepting new connection requests and managing its backlog queue. Alternatively, SHAREPORTWLM can be coded instead; SHAREPORTWLM changes the connection distribution algorithm.

If the same port is reserved for multiple job names, SHAREPORT or SHAREPORTWLM needs to be specified on only one instance of the port reservation. SHAREPORTand SHAREPORTWLM are valid only for TCP ports. The last setting of either SHAREPORT or SHAREPORTWLM is used for all TCP/IP servers that use that port.

SHAREPORTWLM

Required when reserving a port to be shared across multiple listeners on the same interface. When SHAREPORTWLM is specified, TCP/IP allows multiple listeners to listen on the same combination of port and IP address.

The SHAREPORTWLM option can be used instead of SHAREPORT. Like SHAREPORT, SHAREPORTWLM causes incoming connections to be distributed among a set of TCP listeners; however, unlike SHAREPORT, the listener selection is based on WLM server-specific recommendations, modified by the SEF values for each listener. WLM server-specific recommendations are acquired at intervals of approximately 1 minute from the Work Load Manager and reflect the listener's capacity to handle additional work.

If the same port is reserved for multiple job names, SHAREPORT or SHAREPORTWLM needs to be specified on only one instance of the port reservation. SHAREPORT and SHAREPORTWLM are valid only for TCP ports. The last setting of either SHAREPORT or SHAREPORTWLM is used for all TCP/IP servers that use that port.

Result: zAAP and zIIP processor capacity is automatically included when the SHAREPORTWLM parameter is specified and all systems in the sysplex are V1R9 or later.

BIND *ipaddr*

Associates a job name with the IP address, *ipaddr*. When a job with the designated name binds to the IPv4 INADDR_ANY address, or to the IPv6 unspecified address (in6addr_any), the bind is intercepted and converted to a bind to the IP address specified by *ipaddr*. Subsequent bind processing occurs as though the server instance had originally issued the bind to the IP address *ipaddr*.

You can specify either an IPv4 address (in dotted decimal notation) or an IPv6 address (in hexadecimal notation). IPv4-mapped IPv6 addresses and IPv6 addresses with the reserved prefix ::/96 are not supported.

Rule: The BIND *ipaddr* parameter does not apply to the PORTRANGE statement.

Guidelines:

- When you are using the BIND parameter with IPv6 addresses, you should use only manually configured addresses, because autoconfigured addresses might change when the stack is recycled.
- If the IP address specified on the BIND parameter is also specified in a VIPARANGE statement subnet, then VIPARANGE security verification

might occur to determine whether an application can create the dynamic VIPA (DVIPA). For information about security profiles for binding to DVIPAs in the VIPARANGE statement, see z/OS Communications Server: IP Configuration Guide.

SAF *resname*

Indicates that the port is reserved for users that have READ access to the RACF resource

```
EZB.PORTACCESS.sysname.tcpname.resname
```

where

- EZB.PORTACCESS is constant
- *sysname* is the value of the MVS &SYSNAME. system symbol
- *tcpname* is the name of the procedure used to start the TCP stack
- *resname* is the 1-8 character value following the SAF keyword

If the SAF keyword is specified and a user tries to bind to the port and is not allowed access to the resource, the BIND socket call fails.

Tip: The SAF keyword is ignored when VTAM opens a UDP port for Enterprise Extender (EE) network connections. However, it can still be used to prevent other address spaces that are using the same name as the VTAM started task from opening the port.

This is optional and valid for TCP or UDP protocols.

If the *jobname* parameter is specified as an asterisk (*), any user ID that is RACF-permitted to the resource specified by the *resname* value is allowed to bind to the port specified by the value; APF or superuser authority is not required.

This permits multiple users access to the protected port. However, the stack allows only one user to actually BIND to the port at a time. Use SHAREPORT or SHAREPORTWLM to override this behavior for TCP ports.

Guideline: If an application binds to an IP address that is also specified in a VIPARANGE statement subnet, then additional security verification might occur to determine whether the application can create the dynamic VIPA (DVIPA). This additional verification might occur whether the application explicitly binds to the DVIPA address or whether the application binds to the unspecified address and is converted to the DVIPA address using the BIND parameter. For information about security profiles for binding to DVIPAs in the VIPARANGE statement, see z/OS Communications Server: IP Configuration Guide.

NOSMCR

Indicates that Shared Memory Communications over Remote Direct Memory Access (SMC-R) processing is not permitted for TCP connections by using this port. NOSMCR is valid only for TCP ports.

Unreserved Port Options:

SAF *resname*

Indicates that binding to, or listening on, any unreserved port is restricted to users that are permitted to the specified SAF SERVAUTH resource. See the description of the SAF parameter for more information.

DENY

DENY indicates that port access should be denied. DENY can be specified only for unreserved ports (on the PORT UNRSV statement) and only when the specified jobname is an asterisk (*).

A PORT UNRSV *protocol* * DENY statement is needed only if no other PORT UNRSV statements are configured for the specified protocol and you want to prevent all access to unreserved ports using that protocol.

WHENLISTEN

WHENLISTEN indicates that port access control is targeted to TCP applications that are acting as servers (that is, applications able to accept incoming client TCP connections) that issue an explicit bind to a user-specified unreserved port. Permission to use the unreserved port is determined when a TCP listen is issued. If a listen is not issued, no access control check is made. The WHENLISTEN parameter is not available for the UDP protocol, and it is the default for the TCP protocol.

Rule: Every PORT UNRSV statement for the TCP protocol must use the same access control option. You cannot specify, or default to, the WHENLISTEN parameter on some statements and specify the WHENBIND parameter on other statements.

WHENBIND

WHENBIND indicates that permission to use an unreserved port is determined when an explicit bind to a specific local port is issued. This is the default, and only option, for the UDP protocol, and it can affect UDP applications that bind to a specific local port. If the WHENBIND parameter is specified for the TCP protocol, it can affect TCP client applications that bind to a specific local port for outbound connections.

Rule: Every PORT UNRSV statement for the TCP protocol must specify, or default to, the same access control option. You cannot specify the WHENLISTEN parameter on some statements and specify the WHENBIND parameter on other statements.

Steps for modifying

To change a parameter value, you must delete the existing PORT statement by using the DELETE PORT statement, then redefine with the new PORT statement.

Examples

The following example was used for test configuration and is provided here for illustration only. The sample profile, SEZAINST(SAMPPROF), contains the most current assignments.

```
PORT
     7 UDP MISCSERV            ; Miscellaneous Server - echo
     7 TCP MISCSERV            ; Miscellaneous Server - echo
     9 UDP MISCSERV            ; Miscellaneous Server - discard
     9 TCP MISCSERV            ; Miscellaneous Server - discard
    19 UDP MISCSERV            ; Miscellaneous Server - chargen
    19 TCP MISCSERV            ; Miscellaneous Server - chargen
    20 TCP *  NOAUTOLOG        ; FTP Server
;   20 TCP *  NOAUTOLOG SAF FTPDATA ; FTP Server
    21 TCP FTPD1               ; FTP Server
    23 TCP TN3270              ; Telnet 3270 Server
;   23 TCP INETD1 BIND 9.67.113.3 ; z/OS UNIX Telnet server
    25 TCP SMTP                ; SMTP Server
   111 TCP PORTMAP             ; Portmap Server (SUN 3.9)
   111 UDP PORTMAP             ; Portmap Server (SUN 3.9)
```

```
;  111 TCP PORTMAP1           ; Unix Portmap Server (SUN 4.0)
;  111 UDP PORTMAP1           ; Unix Portmap Server (SUN 4.0)
   123 UDP SNTPD              ; Simple Network Time Protocol Server
   135 UDP LLBD               ; NCS Location Broker
   161 UDP OSNMPD             ; SNMP Agent
   389 TCP LDAPSRV            ; LDAP Server
   443 TCP HTTPS              ; http protocol over TLS/SSL
   443 UDP HTTPS              ; http protocol over TLS/SSL
;  500 UDP IKED               ; CS IKE daemon
   512 TCP RXSERVE            ; Remote Execution Server
   514 TCP RXSERVE            ; Remote Execution Server
;  512 TCP * SAF OREXECD      ; z/OS UNIX Remote Execution Server
;  514 TCP * SAF ORSHELLD     ; z/OS UNIX Remote Shell Server
;  515 TCP LPSERVE            ; LPD Server
;  515 TCP AOPLPD             ; Infoprint LPD Server
   520 UDP OMPROUTE           ; OMPROUTE Server (IPv4 RIP)
   521 UDP OMPROUTE           ; OMPROUTE Server (IPv6 RIP)
   580 UDP NCPROUT            ; NCPROUTE Server
   750 TCP MVSKERB            ; Kerberos
   750 UDP MVSKERB            ; Kerberos
   751 TCP ADM@SRV            ; Kerberos Admin Server
   751 UDP ADM@SRV            ; Kerberos Admin Server
; 1700 TCP PAGENT NOAUTOLOG   ; Policy Agent pagentQosListener port
; 1701 TCP PAGENT NOAUTOLOG   ; Policy Agent pagentQosCollector port
  3000 TCP CICSTCP            ; CICS Socket
  3389 TCP MSYSLDAP           ; LDAP Server for Msys
; 4159 TCP NSSD               ; CS NSS daemon
; 4500 UDP IKED               ; CS IKE daemon
;16310 TCP PAGENT NOAUTOLOG   ; Policy Agent server listener port
;
```

The following examples control application access to unreserved ports:

- To deny all TCP explicit binds to an unreserved port, add the following statement to your profile:

  ```
  PORT  UNRSV  TCP  *  DENY WHENBIND
  ```

- To allow TCP explicit binds to an unreserved port but deny all TCP listens on an unreserved port, add the following statement to your profile:

  ```
  PORT  UNRSV  TCP  *  DENY  WHENLISTEN
  ```

- To deny all TCP listens on an unreserved port, except for applications that match *jobname* value ABC*, add the following statement to your profile:

  ```
  PORT  UNRSV  TCP  ABC*   WHENLISTEN
  ```

  Guideline: If the ports that applications ABC* are accessing are predictable, you should use PORT reservation statements for those specific ports instead of using the PORT UNRSV statement.

- To deny all TCP listens on an unreserved port, except for application MYAPP1 and all users permitted to EZB.PORTACCESS.*sysname.tcpname*.GENERIC, add the following statements to your profile:

  ```
  PORT  UNRSV  TCP  MYAPP1
  PORT  UNRSV  TCP  *  SAF  GENERIC
  ```

- To deny all UDP explicit binds to an unreserved port, except for users permitted to EZB.PORTACCESS.*sysname.tcpname*.GENERIC, add the following statement to your profile:

  ```
  PORT  UNRSV  UDP  *  SAF  GENERIC
  ```

Usage notes

- If there are no PORT UNRSV statements configured for this stack, any user can use a port that is not reserved in this list or that is not reserved with the PORTRANGE statement. If you have TCP/IP hosts in your network that use ports in the range 1 - 1023 for privileged applications, you should reserve them with this statement, the PORTRANGE statement, or the RESTRICTLOWPORTS parameter on the TCPCONFIG or UDPCONFIG statements.

- If an application attempts to access a specific port by explicitly binding for UDP, by explicitly binding, or listening for TCP, and no PORT or PORTRANGE statement is found that matches that port and protocol (that is, the port is unreserved for that protocol), then a check is made for PORT UNRSV statements. The following list shows the possible results:
  - If there are no PORT UNRSV statements for that protocol, the access is allowed.
  - If there are any PORT UNRSV statements for the protocol, a search is made for the most specific match to the application's job name.
    - If a match is found, the access is allowed unless the closest matching PORT UNRSV statement contains the DENY keyword, or if it contains the SAF keyword and the user is not permitted to the specified SAF resource.
    - If no matching PORT UNRSV statement is found, the access is denied.
- For z/OS UNIX applications, you can reserve a port by specifying the job name of the application or you can use the name of the started JCL procedure for the z/OS UNIX kernel address space to enable any application (except applications using the Pascal API) to bind to the port. This name is typically OMVS unless a different name is explicitly specified in the STARTUP_PROC parameter in the BPXPRMxx parmlib member. See z/OS MVS Initialization and Tuning Reference for more details about the STARTUP_PROC parameter.
- For syslogd, you must include the following PORT statement:

  ```
  PORT
      514 UDP OMVS        ; syslogd Server
  ```

  This port is required for syslogd to accept log data from remote syslogd servers.

  Guideline: Instead of OMVS, you can also use the job name of the syslog daemon on this port reservation statement. If your syslog daemon's job name is SYSLOGD1, you can specify:

  ```
  PORT   514   UDP   SYSLOGD1
  ```
- If you want SNMP OSA Management support, see z/OS Communications Server: IP Configuration Guide for more information about the PORT statement.
- The NOSMCR option is enforced during TCP bind() processing. To allow servers that bind to a port that is configured with the NOSMCR option to use SMC-R communications, you need to perform the following steps:
  1. Delete the existing port reservation by using the VARY TCPIP,,OBEYFILE command with a data set that contains a DELETE PORT statement.
  2. Create a reservation for the port by using the VARY TCPIP,,OBEYFILE command with a data set that contains a PORT statement without the NOSMCR parameter.
  3. Stop and restart the servers that use the port.

Related topics

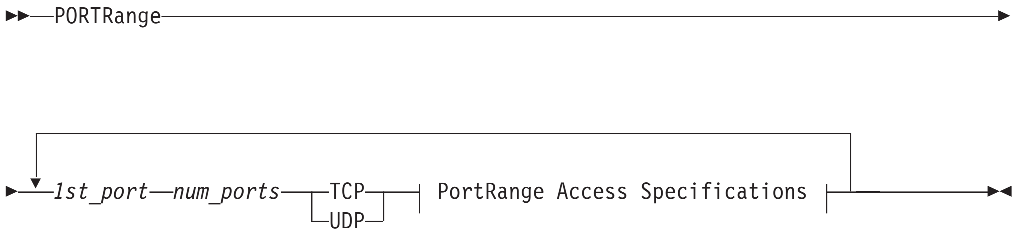
- "AUTOLOG statement" on page 23
- "DELETE statement" on page 43
- "PORTRANGE statement" on page 266
- "TELNETPARMS statements" on page 587

PORTRANGE statement

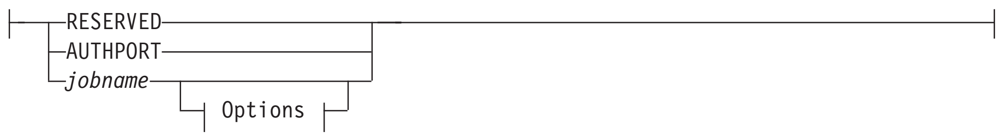
Use the PORTRANGE statement to reserve a range of ports for specified user IDs, procedures, or job names. The PORTRANGE statement can also specify other options that apply to all ports in the range.

Rule: The portrange options (NOAUTOLOG, DELAYACKS, and so on) must be specified in the same order as they appear on the following syntax diagram.

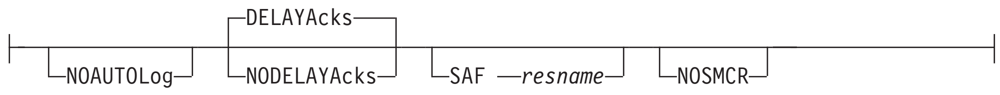
Syntax



PortRange Access Specifications:



Options:



Parameters

`1st_port`
: The starting port for a range of ports to reserve. The same port number cannot appear in multiple PORTRANGE statements, nor can the port be specified on both PORTRANGE and PORT statements. If the port is specified on a PORT statement prior to this statement, this port range is ignored. If the port is specified on a PORT statement that follows this statement, the port in the PORT statement is ignored. An error message is generated in either case. *1st_port* is a value in the range 1 - 65535.

  If the *1st_port* and *num_ports* values that are specified result in a range of ports that exceeds the maximum port number of 65535, the ports up to 65535 are reserved and those greater than 65535 are ignored.

`num_ports`
: The number of ports to reserve. The ports reserved cannot overlap other ranges specified by a PORTRANGE statement. No ports within this range can be specified on a PORT statement. If the port is specified on a PORT statement prior to this statement, this port range is ignored. If the port is specified on a

PORT statement that follows this statement, the port in the PORT statement is ignored. An error message is generated in either case. *num_port* is a value in the range 1 - 65535.

If the *1st_port* and *num_ports* values that are specified result in a range of ports that exceeds the maximum port number of 65535, the ports up to 65535 are reserved and those greater than 65535 are ignored.

`jobname`

The MVS job name that can use the port. You can specify the *jobname* value as 1 - 8 characters, an asterisk (*) wildcard value, or a 1 - 7 character prefix followed by an asterisk wildcard value. Specify an asterisk as the *jobname* value to reserve a port without specifying a particular job name. This is useful when you do not know the exact job name or when you want to allow several different applications to serially bind to the port. Specify a 1 - 7 character prefix followed by an asterisk to enable all job names that match the prefix to access the ports in the range.

Restrictions:

- For UDP, only one job name can be associated with a port.
- To reserve a port that is to be monitored by the AUTOLOG function, the *jobname* value must exactly match the *jobname* value on the AUTOLOG statement; you cannot use an asterisk wildcard value.

Guideline: If a TCP port is to be shared by multiple users, use the PORT statement instead. The PORTRANGE statement does not support sharing of ports.

Determining the job name to be associated with a particular client or server application depends on the environment in which the application is run.

- Applications run from batch use the batch job name.
- Applications started from the MVS operator console use the started procedure name as the job name.
- Applications run from a TSO user ID use the TSO user ID as the job name.
- Applications run from the z/OS shell normally have a job name that is the logged on user ID plus a 1-character suffix.
- Authorized users can run applications from the z/OS shell and use the _BPX_JOBNAME environment variable to set the job name. In this case, the value specified for the environment variable is the job name.
- Use the name of the started JCL procedure for the UNIX System Services kernel address space to enable any application (except for applications using the Pascal API) to bind to the port. This name is typically OMVS unless a different name is explicitly specified in the STARTUP_PROC parameter in the BPXPRMxx parmlib member.
- To reserve the port and not allow any application access to it, use the name RESERVED.
- To reserve ports for the FTP server's use as passive data ports, use the name AUTHPORT and the protocol TCP. You must also code the PASSIVEDATAPORTS value in the FTP server's FTP.DATA data set.
- Use the name of the VTAM started task for the UDP ports that are to be used for Enterprise Extender (EE) network connections.

  Restriction: The VTAM jobname can NOT include a wildcard character (*) when it reserves EE UDP ports.

RESERVED
Indicates that all ports in the port range are not available for use by any user.

AUTHPORT
Indicates that all ports in the port range are not available for use by any user except FTP, and only when FTP is configured to use PASSIVEDATAPORTS. AUTHPORT is valid only with the TCP protocol.

NOAUTOLOG
Tells the TCP/IP address space *not* to restart the server if it was stopped previously. Otherwise, the default is to restart the server if it was stopped previously.

DELAYACKS | NODELAYACKS

NODELAYACKS
Specifies that an acknowledgment is returned immediately when a packet is received with the PUSH bit on in the TCP header. The NODELAYACKS parameter on the PORTRANGE statement, affects only connections that use this port. Specifying the NODELAYACKS parameter on the PORTRANGE statement overrides the specification of the DELAYACKS parameter on the TCP/IP stack TCPCONFIG profile statement, or on any of the following statements used to configure the route used by the TCP connection:

- The TCP/IP stack BEGINROUTES or GATEWAY profile statements
- The Policy Agent RouteTable statement
- The OMPROUTE configuration statements

DELAYACKS
Delays transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. The DELAYACKS parameter on the PORTRANGE statement affects only connections that use this port. This is the default, but the behavior can be overridden by specifying the NODELAYACKS parameter on the TCP/IP stack TCPCONFIG profile statement, or on any of the following statements used to configure the route used by the TCP connection:

- The TCP/IP stack BEGINROUTES or GATEWAY profile statements
- The Policy Agent RouteTable statement
- The OMPROUTE configuration statements

SAF *resname*
SAF *resname* indicates that all ports in the range are reserved for users that have READ access to the RACF resource.

```
EZB.PORTACCESS.sysname.tcpname.resname
```

where

- EZB.PORTACCESS is constant
- *sysname* is the value of the MVS &SYSNAME. system symbol
- *tcpname* is the name of the procedure used to start the TCP stack
- *resname* is a 1-8 character value following the SAF keyword

If the SAF keyword is specified and an application tries to bind to a port in the port range, and the user ID associated with the application is not permitted to the resource, the BIND socket call fails.

This is optional and valid for TCP or UDP protocols.

If the *jobname* value is specified as an asterisk (*), any user ID that is RACF-permitted to the resource specified by the *resname* value is allowed to bind to the port; APF or superuser authority is not required.

Guideline: If an application binds to an IP address that is also specified in a VIPARANGE statement subnet, then additional security verification might occur to determine whether the application can create the dynamic VIPA (DVIPA). For information about security profiles for binding to DVIPAs in the VIPARANGE statement, see z/OS Communications Server: IP Configuration Guide

NOSMCR
Indicates that Shared Memory Communications over Remote Direct Memory Access (SMC-R) processing is not permitted for TCP connections by using any port in this range. NOSMCR is valid only for TCP ports.

Steps for modifying

To change a parameter value, you must delete the existing PORTRANGE statement by using the DELETE PORTRANGE statement, then redefine the parameter with the new PORTRANGE statement.

Examples

This example shows a PORTRANGE statement used to reserve a large number of ports for a single test system.

```
PORTRANGE
    4000 200  TCP TESTSYS
```

The following example shows a PORTRANGE statement that reserves port 111 for both UDP and TCP for one user, ports 500 - 504 for two different users, one using UDP and one using TCP, and ports 700 - 703 for TCP users with job names that begin with the prefix ABCD.

```
PORTRANGE
     111   1  UDP  PORTMAP
     111   1  TCP  PORTMAP
     500   5  UDP  USER1
     500   5  TCP  USER2
     700   4  TCP  ABCD*
```

Usage notes

- A range of ports specified in a VARY TCPIP,,OBEYFILE command data set are added to the list of ports already reserved.
- Any user can use a port that is not reserved by a PORT or PORTRANGE statement. If you have TCP/IP hosts in your network that reserve ports in the range 1 - 1023 for privileged applications, you should reserve them either with this statement, the PORT statement, or the RESTRICTLOWPORTS parameter on the TCPCONFIG or UDPCONFIG statements.
- If you are reserving ports for the INADDRANYPORT() parameter in the BPXPRMxx SYS1.PARMLIB member, you must specify the name of the started JCL procedure for the z/OS UNIX kernel address space to enable any application (except for applications using the Pascal API) to bind to the port. This name is typically OMVS unless a different name is explicitly specified in the STARTUP_PROC parameter in the BPXPRMxx parmlib member. See z/OS MVS Initialization and Tuning Reference for more details about the STARTUP_PROC parameter. You can use IBM Health Checker for z/OS

enhancements to check whether the range of ports specified by the INADDRANYPORT and INADDRANYCOUNT parameter of the BPXPRMxx parmlib member is reserved for OMVS on the TCP/IP stack when operating in a CINET environment. For more details about IBM Health Checker for z/OS enhancements, see the IBM Health Checker for z/OS enhancements information in the z/OS Communications Server: IP Diagnosis Guide

- The NOSMCR option is enforced during TCP bind() processing. To allow servers that bind to a port in this range that is configured with the NOSMCR option to use SMC-R communications, you need to perform the following steps:
  1. Delete the existing port reservations by using the VARY TCPIP,,OBEYFILE command with a data set that contains a DELETE PORTRANGE statement.
  2. Create reservations for the port by using the VARY TCPIP,,OBEYFILE command with a data set that contains a PORTRANGE statement without the NOSMCR parameter.
  3. Stop and restart the servers that use the ports.

Related topics

- “DELETE statement” on page 43
- “PASSIVEDATAPORTS (FTP server) statement” on page 824
- “PORT statement” on page 257

PRIMARYINTERFACE statement

Restriction: The PRIMARYINTERFACE statement applies to IPv4 only.

Use the PRIMARYINTERFACE statement to specify which interface is to be designated as default local host for use by the GETHOSTID() function.

The PRIMARYINTERFACE statement's IP address is not used as the source IP address for any out-going datagrams, unless that same address is configured as the SOURCEVIPA address.

| If no HOME statements are defined in the profile, PRIMARYINTERFACE defaults
| to the first IPv4 INTERFACE statement listed in the profile.

Syntax

```
►►──PRImaryinterface ──interface_name──────────────────────────────────────►◄
```

Parameters

interface_name
: The name of an interface that is to be the primary interface. This interface must have already been defined to TCP/IP. If you specify the name of a dynamic VIPA interface, the dynamic VIPA must have been defined in a VIPADYNAMIC block. You cannot specify a loopback interface name.

Steps for modifying

To modify parameters for the PRIMARYINTERFACE statement, you must respecify the statement with the new parameters.

Include the PRIMARYINTERFACE statement in a VARY TCPIP,,OBEYFILE command data set under the following conditions:

- Your primary interface is defined by the DEVICE, LINK, and HOME profile statements.
- The VARY TCPIP,,OBEYFILE command data set includes a HOME statement.
- You want to preserve your primary interface settings.

If you do not include the PRIMARYINTERFACE statement under these conditions, the primary interface is reset to the first entry in the new HOME list. If your primary interface is defined by an INTERFACE profile statement, including a HOME statement in a VARY TCPIP,,OBEYFILE command data set does not affect the primary interface settings.

Examples

This example shows a PRIMARYINTERFACE statement specifying a token-ring:

```
PRIMARYINTERFACE TR1
```

You can verify which HOME entry is primary by using the Netstat HOME/**-h** command:

```
Home address list:
Address          Link            Flg
9.67.113.61      TR1             P
9.67.116.125     CTCD00
127.0.0.1        LOOPBACK
```

Usage notes

- Because of the way dynamic VIPA links are added to the TCP/IP stack, you cannot specify a PRIMARYINTERFACE statement for a dynamic VIPA link in the same Profile data set as the VIPADYNAMIC block that defines the dynamic VIPA link. This is true for the initial Profile data set or a Profile data set specified on a VARY TCPIP,,OBEYFILE command. In order to specify a dynamic VIPA link on a PRIMARYINTERFACE statement, the dynamic VIPA link must have been defined to the stack in a previous Profile data set.
- The primary interface is flagged in the Netstat HOME/**-h** display.
- If the PRIMARYINTERFACE statement is not specified, then the first address in the HOME list is designated as the default local host.

Related topic

- “HOME statement” on page 136

SACONFIG statement

Use the SACONFIG statement to configure the SNMP TCP/IP subagent. If the SACONFIG statement is not specified, the subagent is started by TCP/IP initialization but SNMP SET support is disabled.

Syntax

Tip: Specify the parameters for this statement in any order.

```
                ┌──────────────────────────────────────────┐
                ▼                                          │
►►──SACONFig──┬──────────────────────────────────────────┬─┴────────────────────►◄
              │  ┌─AGENT 161──────────────────┐           │
              ├──┤                            ├───────────┤
              │  └─AGENT──agent_port_number───┘           │
              │  ┌─COMMUNity public─────────────────┐     │
              ├──┤                                  ├─────┤
              │  └─COMMUNity──community_string──────┘     │
              │  ┌─ENABLed──┐                             │
              ├──┤          ├─────────────────────────────┤
              │  └─DISABLed─┘                             │
              │  ┌─OSADISabled─┐                          │
              ├──┤             ├──────────────────────────┤
              │  └─OSAENabled──┘                          │
              ├───────────────────────────────────────────┤
              │  └─OSASF osasf_port_number─┘              │
              │  ┌─SACACHETime 30──────────┐              │
              ├──┤                         ├──────────────┤
              │  └─SACACHETime cache_time──┘              │
              │  ┌─SETSDISabled─┐                         │
              └──┤              ├─────────────────────────┘
                 └─SETSENAbled──┘
```

Parameters

AGENT

A port number in the range 1 - 65 535 used in establishing communication with the SNMP agent. For the TCP/IP SNMP subagent to communicate with the z/OS Communications Server SNMP agent, the port number specified must match the port number specified on the -p parameter when the SNMP agent is started. The default value is 161 when processing the initial profile only. If SACONFIG is specified in a VARY TCPIP,,OBEYFILE command data set without the AGENT parameter, the value is unchanged.

COMMUNITY

A 1 - 32 character string used as the community name (or password) in establishing contact with the SNMP agent. It is not converted to uppercase by profile processing. It cannot contain any imbedded white space or control characters (such as blank, tab, end of line, or end of file) and cannot contain any imbedded semicolons (semicolons are treated as comment delimiters). For the TCP/IP SNMP subagent to communicate with the z/OS Communications Server SNMP agent, the community name specified on the COMMUNITY keyword must match one that is defined in the PW.SRC or SNMPD.CONF data set used by the SNMP agent or specified on the -c parameter when the SNMP agent is started.

Restriction: The community name is case sensitive.

For more information about how the community name is used to permit access to the SNMP agent, see Step 1: Configure the SNMP agent (OSNMPD), in z/OS Communications Server: IP Configuration Guide.

The default value is *public* when processing the initial profile only. If SACONFIG is specified in a VARY TCPIP,,OBEYFILE command data set without the COMMUNITY parameter, the value is unchanged.

DISABLED

If specified in PROFILE.TCPIP at initialization, indicates that the SNMP subagent should not be started. Specify this parameter if you do not require any of the SNMP MIB data supported by the TCP/IP subagent. By default, the SNMP subagent is started by TCP/IP initialization.

If specified in a VARY TCPIP,,OBEYFILE command data set, indicates that the currently active subagent task should be terminated.

SNMP MIB objects supported by the z/OS Communications Server SNMP agent and subagents other than the TCP/IP SNMP subagent are still available. For information about which MIB objects are supported by the SNMP agent and subagent, see the z/OS Communications Server: IP User's Guide and Commands.

ENABLED
Indicates that the SNMP subagent should be started at the completion of the initial profile processing, or of VARY TCPIP,,OBEYFILE command processing.

OSADISABLED
Indicates that OSA Management support is not required at this TCP/IP instance. If this support was previously enabled, then specifying this parameter disables the support.

OSAENABLED
Indicates that OSA Management support is required at this TCP/IP instance. For optimal performance, specify OSAENABLED only at the TCP/IP instance from which Management support is needed. By default, OSA data retrieval is not enabled.

The SNMP subagent must be enabled, as it provides support for retrieval of SNMP management data about OSA devices and links. Therefore, do not specify the DISABLED parameter for this TCP/IP instance.

To retrieve the data, there must also be at least one TCP/IP instance active for which the OSASF parameter and its port number have been specified in the SACONFIG statement.

OSASF *osasf_port_number*
A value between 0 and 65 535. There is no default. A value of 1 - 65 535 indicates a port number and marks the corresponding TCP/IP instance as a candidate to communicate with OSA/SF for retrieval of SNMP management data regarding ATM devices and links. A value of 0 indicates that the corresponding TCP/IP instance is no longer a candidate to communicate with OSA/SF, in the event that the OSA/SF-to-TCP/IP connection is restarted.

Guideline: When multiple TCP/IP instances specify that OSA management data retrieval is wanted, it is suggested that all be configured with the same OSASF parameter. Only one TCP/IP instance connects directly to OSA/SF. Other instances connect to OSA/SF using this primary TCP/IP instance.

SACACHETIME *cache_time*
The number of seconds (in the range 0 - 3 600) that the TCP/IP subagent caches management data. If a request for management data is received and the amount of time specified by the *cache_time* value has expired, the TCP/IP subagent retrieves a new copy of the management data from the TCP/IP stack. A value of 0 indicates that the TCP/IP subagent should not cache any data, but always retrieve the current value of the data from the TCP/IP stack. The default value is 30 seconds when processing the initial profile only. The subagent's *cache_time* value can also be changed by an SNMP SET request.

SETSDISABLED
Indicates that the SNMP subagent should not process SNMP SET requests.

SETSENABLED
Indicates that the SNMP subagent should process SNMP set requests. For example, SETSENABLED allows a user who issued an SNMP set request to cancel connections and start and stop devices using the SNMP agent security instead of RACF security. By default, the processing of SNM P SET requests is disabled.

Steps for modifying

To modify parameters for the SACONFIG statement, you must respecify the statement with the new parameters.

- If you modify the community name or Agent port number, you must recycle the TCP/IP subagent or the SNMP Agent for the changes to take effect.
- If you modify the OSA/SF port number, you must recycle the TCP/IP subagent or the OSA/SF IOASNMP application for the changes to take effect.

Examples

```
SACONFIG COMMUNITY USACCESS AGENT 528
SACONFIG DISABLED
SACONFIG SETSENABLED OSAENABLED OSASF 2026
```

Usage notes

When you specify more than one SACONFIG statement in the initial profile data set, or in a data set referenced by the VARY TCPIP,,OBEYFILE command, the default value ENABLED is set even if the DISABLED value was specified in a previous SACONFIG statement.

SMFCONFIG statement

Use the SMFCONFIG statement to provide SMF logging for Telnet, FTP, IPSec, TCP/IP API, TCP/IP stack, and Shared Memory Communications over Remote Direct Memory Access (SMC-R) activity.

Using SMFCONFIG to turn on SMF logging allows you to request that standard subtypes are assigned to the TCP/IP SMF records. The SMFPARMS statement provides a similar capability but requires the installation to select the subtype numbers to be used. Use the SMFCONFIG statement instead of SMFPARMS. See the information about accounting for SMF records in z/OS Communications Server: IP Configuration Guide.

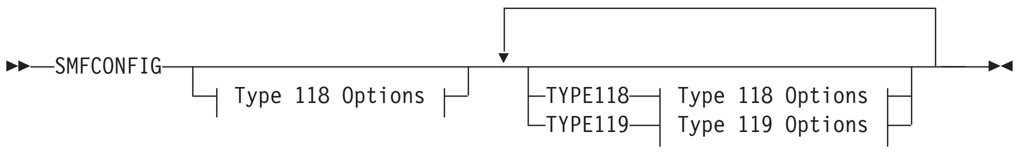
The SMFCONFIG profile statement controls only whether SMF records are written to the MVS SMF data sets. Some of the SMF 119 records are also available to applications that connect to the following network management interface (NMI) services:

- Real time TCP connection SMF data NMI (SYSTCPCN)
- Real time SMF data NMI (SYSTCPSM)

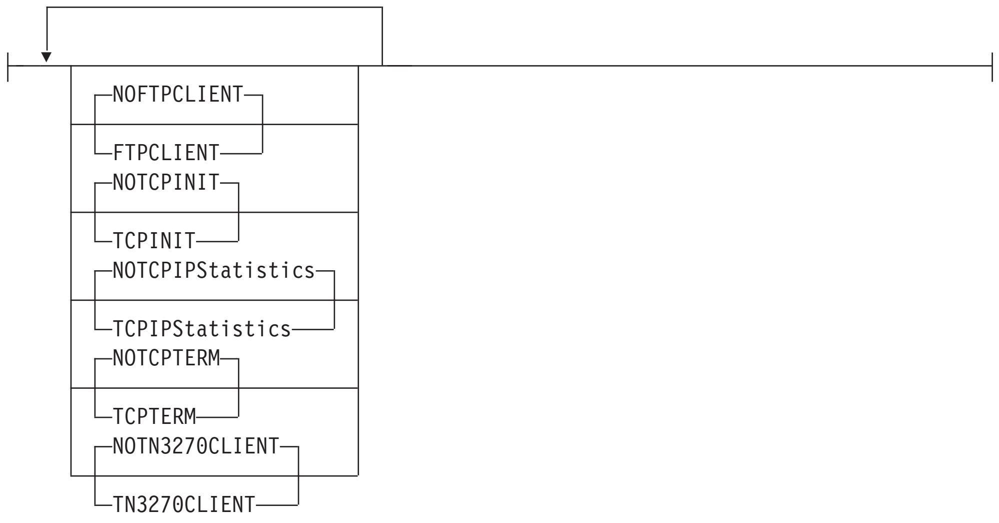
These functions are part of the real time TCP/IP network monitoring NMI. For more information about the real time TCP/IP network monitoring NMI functions,see “NETMONITOR statement” on page 234. If you want your application to process only SMF 119 records by using the real time TCP/IP network monitoring NMI functions, you need to configure only the NETMONITOR profile statement. You do not need to request support for these records on the SMFCONFIG profile statement.

Syntax

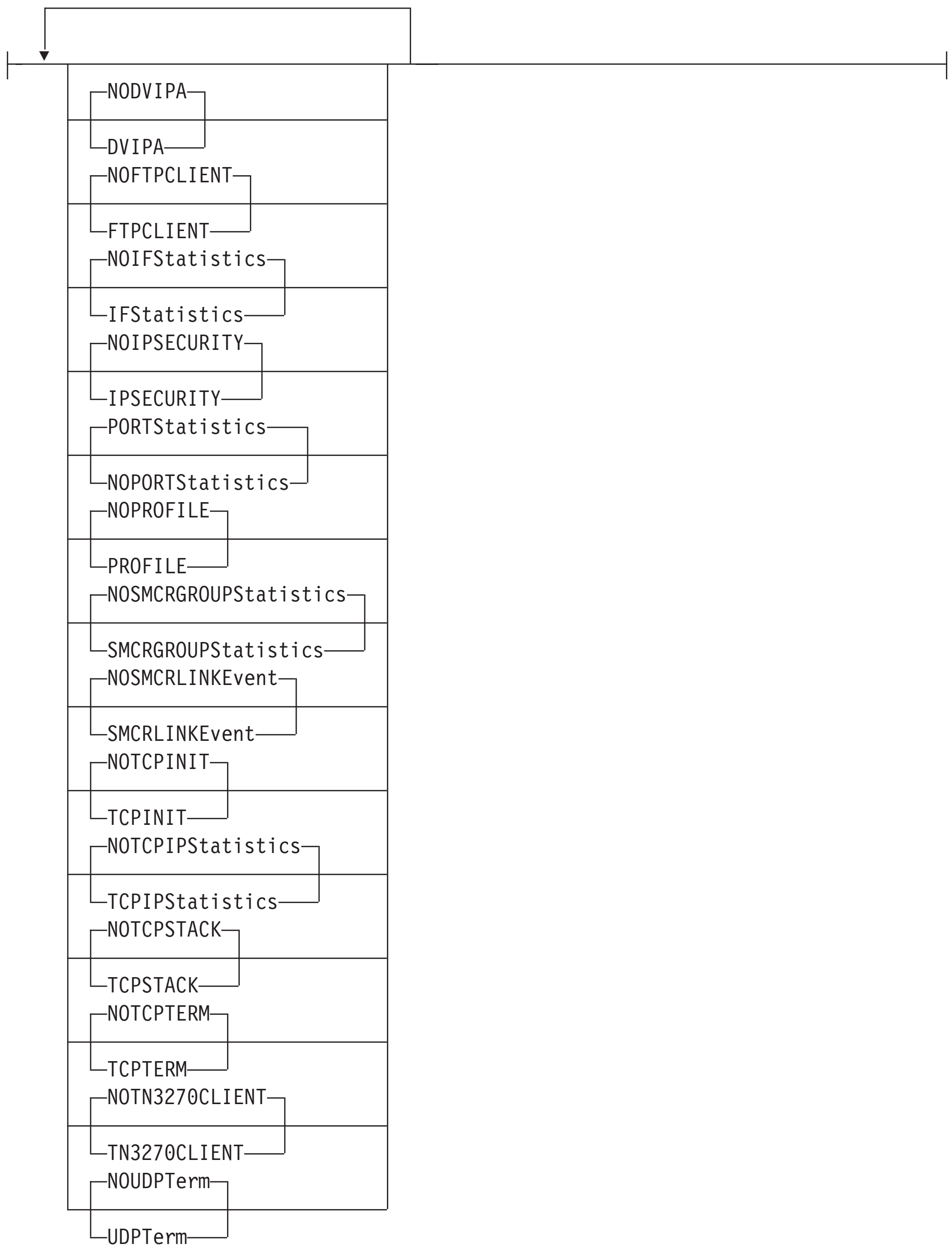
Tip: Specify the parameters for this statement in any order.



Type 118 Options:



Type 119 Options:



Parameters

DVIPA | NODVIPA

NODVIPA
Requests that SMF records of subtype 32, 33, 34, 35, 36, and 37 not be created when various sysplex events occur. This operand is valid if the current record type setting is TYPE119. This is the default value.

DVIPA
Requests that SMF records of subtype 32, 33, 34, 35, 36, and 37 be created when various sysplex events occur. This operand is valid if the current record type setting is TYPE119.

FTPCLIENT | NOFTPCLIENT

NOFTPCLIENT
Requests that SMF records of subtype 3 not be created when a user invokes the FTP client command. The record format affected (Type 118 or Type 119) by this operand is determined by the most recently specified setting of the TYPE118 or TYPE119 operand. This is the default value.

FTPCLIENT
Requests that SMF records of subtype 3 be created when a user invokes the FTP client command. The record format affected (Type 118 or Type 119) by this operand is determined by the most recently specified setting of the TYPE118 or TYPE119 operand.

`IFSTATISTICS | NOIFSTATISTICS`

NOIFSTATISTICS
Requests that SMF type 119 records of subtype 6 and subtype 44 not be created. This operand is valid if the current record type setting is TYPE119. This is the default value.

IFSTATISTICS
Requests that SMF type 119 records that are related to interface statistics are created. The following record subtypes are created:

- Subtype 6 records that contain statistics that are related to interface utilization
- Subtype 44 records that contain statistics that are related to 10GbE RoCE Express interface utilization for SMC-R communications

Note that these records are created periodically based on the SMF interval in effect. This operand is valid if the current record type setting is TYPE119.

`IPSECURITY | NOIPSECURITY`

NOIPSECURITY
SMF type 119 records of subtypes 77, 78, 79, and 80 are not created. This operand is valid if the current record type setting is TYPE119. This is the default value.

IPSECURITY
Creates SMF type 119 records of subtypes 77 and 78 when a dynamic tunnel is added and removed. Creates SMF type 119 records of subtypes 79 and 80 when a manual tunnel is activated or deactivated. This operand is valid if the current record type setting is TYPE119.

`PORTSTATISTICS | NOPORTSTATISTICS`

NOPORTSTATISTICS
Requests that SMF type 119 records of subtype 7 not be created. This operand is valid if the current record type setting is TYPE119. This is the default value.

PORTSTATISTICS
Requests that SMF type 119 records of subtype 7 containing statistics related to reserved PORT utilization be created. Note that these records are created periodically based on the SMF interval in effect. This operand is valid if the current record type setting is TYPE119.

`PROFILE| NOPROFILE`

NOPROFILE

Requests that SMF type 119 records of subtype 4 not be created. This operand is valid if the current record type setting is TYPE119. This is the default value.

Results: If SMFCONFIG PROFILE is in effect, and then SMFCONFIG NOPROFILE is specified in a profile data set referenced by the VARY TCPIP,,OBEYFILE command, one final SMF type 119 record of subtype 4 is created to record the fact that this function has been turned off and no more SMF records of this subtype are created.

PROFILE

Requests that SMF type 119 records of subtype 4 be created. These records are SMF event records that provide TCP/IP stack profile information. They are created when the stack is first started and when profile changes occur as a result of VARY TCPIP,,OBEYFILE command processing. This operand is valid if the current record type setting is TYPE119.

SMCRGROUPSTATISTICS | NOSMCRGROUPSTATISTICS

SMCRGROUPSTATISTICS

Requests that SMF type 119 records of subtype 41 not be created. This operand is valid if the current record type setting is TYPE119. This is the default value.

NOSMCRGROUPSTATISTICS

Requests that SMF type 119 records of subtype 41 containing statistics related to SMC-R link groups be created. These records are created periodically based on the SMF interval in effect. This operand is valid if the current record type setting is TYPE119.

SMCRLINKEVENT | NOSMCRLINKEVENT

SMCRLINKEVENT

Requests that SMF type 119 records of subtype 42 and 43 not be created. This operand is valid if the current record type setting is TYPE119. This is the default value.

NOSMCRLINKEVENT

Requests that SMF type 119 records of subtype 42 and 43 be created. The SMF records of subtype 42 are created when SMC-R links are started, and the SMF records of subtype 43 are created when SMC-R links are ended. This operand is valid if the current record type setting is TYPE119.

TCPINIT | NOTCPINIT

NOTCPINIT

Requests that SMF records of subtype 1 not be created when TCP connections are established. The record format affected (Type 118 or Type 119) by this operand is determined by the most recent setting of the TYPE118 or TYPE119 operand. This is the default value.

TCPINIT

Requests that SMF records of subtype 1 be created when TCP connections are established. The record format collected (Type 118 or Type 119) is determined by the most recently specified TYPE118 or TYPE119 operand.

TCPIPSTATISTICS | NOTCPIPSTATISTICS

NOTCPIPSTATISTICS
Requests that SMF records of subtype 5 not be created. The record format affected (Type 118 or Type 119) by this operand is determined by the most recently specified setting of the TYPE118 or TYPE119 operand. This is the default value.

TCPIPSTATISTICS
Requests that SMF records of subtype 5 containing TCP/IP statistics be created. Note that these records are created periodically based on the SMF interval in effect. The record format collected (Type 118 or Type 119) is determined by the most recent setting of the TYPE118 or TYPE119 operand.

TCPSTACK | NOTCPSTACK

NOTCPSTACK
Requests that SMF type 119 records of subtype 8 not be created. This operand is valid if the current record type setting is TYPE119. This is the default value.

TCPSTACK
Requests that SMF type 119 records of subtype 8 be created when a TCP stack is activated and when it is terminated. This operand is valid if the current record type setting is TYPE119.

TCPTERM | NOTCPTERM

NOTCPTERM
Requests that SMF records of subtype 2 not be created when TCP connections are terminated. The record format affected (Type 118 or Type 119) by this operand is determined by the most recently specified setting of the TYPE118 or TYPE119 operand. This is the default value.

TCPTERM
Requests that SMF records of subtype 2 be created when TCP connections are terminated. The record format collected (Type 118 or Type 119) is determined by the most recently specified TYPE118 or TYPE119 operand.

TN3270CLIENT | NOTN3270CLIENT

NOTN3270CLIENT
Requests that SMF type 118 records of subtype 4, or type 119 records of subtype 22 or 23 not be created. The record format affected (Type 118 or Type 119) by this operand is determined by the most recently specified setting of the TYPE118 or TYPE119 operand. This is the default value.

TN3270CLIENT
Requests that SMF type 118 records of subtype 4, or type 119 records of subtype 22 and 23 be created when the TSO Telnet Client code initiates or terminates a connection (respectively for type 119). The record format collected (Type 118 or Type 119) is determined by the most recently specified TYPE118 or TYPE119 operand.

UDPTERM | NOUDPTERM

NOUDPTERM
Requests that SMF type 119 records of subtype 10 not be created. This operand is valid if the current record type setting is TYPE119. This is the default value.

UDPTERM
Requests that SMF type 119 records of subtype 10 be created when a UDP Socket is closed. This operand is valid if the current record type setting is TYPE119.

Steps for modifying

To modify parameters for the SMFCONFIG statement, you must respecify the statement with the new parameters.

VARY TCPIP,,OBEYFILE command processing does not reset previous settings to the default.

Statement dependency

- Use of SMFCONFIG is preferable to SMFPARMS to standardize subtypes. If SMFPARMS is encountered after an SMFCONFIG statement, an error message is displayed and the SMFPARMS parameters are ignored. If SMFCONFIG is not coded, no SMF records are logged (assuming that SMFPARMS is not coded either).
- SMFPARMS is valid only for Type 118 records. Type 119 records have default subtype values that are not installation-configurable.

Examples

This example requests SMF records for TCP connection initialization, TCP connection termination, FTP client, Telnet client, and TCP/IP statistics:

```
SMFCONFIG TCPINIT TCPTERM FTPCLIENT TN3270CLIENT TCPIPSTATISTICS
```

The format type default is TYPE118. If you use SMFCONFIG to activate SMF recording, you do not need to make any changes to continue receiving the same recording. If you want to use the new records, specify TYPE119, followed by any of the SMF records that you want.

For example, if the following is specified:

```
SMFCONFIG FTPCLIENT TN3270CLIENT
          TYPE119 FTPCLIENT TN3270CLIENT
```

The recording is Type 118 FTP, TN3270 client records and Type 119 FTP, TN3270 client records.

Usage notes

Requirement: SMF must be active and properly configured to allow the recording of Type 118 or Type 119 records, depending on which types are being used by the configuration.

Tip: The TYPE118 keyword can be omitted when designating Type 118 options as long as they are specified before the Type 119 options.

Related topic

- "NETMONITOR statement" on page 234

SMFPARMS statement

Use the SMFPARMS statement to log the use of TCP by applications using SMF Type 118 log records. You can log Telnet and FTP client activity, and TCP API activity.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──SMFPARMS──inittype──termtype──clienttype──────────────────────────────►◄
```

Parameters

inittype
: An integer in the range 0 - 255 specifying the subtype field in the API initialization records. The value 0 indicates that no API initialization is written.

termtype
: An integer in the range 0 - 255 specifying the subtype field in the API termination records. The value 0 indicates no API termination records are written.

clienttype
: An integer in the range 0 - 255 specifying the subtype field in the FTP or Telnet client. The value 0 indicates that no FTP or Telnet client records are written.

Steps for modifying

To modify parameters for the SMFPARMS statement, you must respecify the statement with the new parameters.

Statement dependency

SMFPARMS is valid only for Type 118 SMF records. Type 119 records have default subtype values that are not installation-configurable. As such, the only way to activate the recording of Type 119 records is by using SMFCONFIG.

Examples

- Either of the following statements would produce API initialization and termination records but no FTP or Telnet client records:

```
SMFPARMS    3   4 0
SMFPARMS    3   4
```

- The following statement would produce client records only:

```
SMFPARMS    0   0   5
```

- Because one of the parameters is missing, this statement would generate an error and not produce any records:

```
SMFPARMS  3
```

Usage notes

- The values for each subtype should be unique.
- If *inittype*, *termtype*, or *clienttype* have the value of 0, no attempt is made to write the respective record.

- The format of the log information differs for Telnet and FTP client activity, and TCP API activity.

SOMAXCONN statement

Use the SOMAXCONN statement to specify the maximum number of connection requests queued for any listening socket.

Syntax

Rule: Specify the parameters in the order shown here.

```
      ┌─SOMAXCONN 1024──────────────────────┐
►►────┼─────────────────────────────────────┼─────────────────────────────────────►◄
      └─SOMAXCONN ──maximum_queue_depth─────┘
```

Parameters

maximum_queue_depth
: The maximum number of pending connection requests queued for any listening socket. The minimum value is 1, the maximum value is 2147483647, and the default value is 1024.

  This number is stored as a fullword integer, but most implementations of TCP/IP hardcode a value in the range 5 - 10.

  This number is the maximum depth for any listening stream socket, but you can specify a shorter queue length when the listen performed for the socket.

Steps for modifying

To modify parameters for the SOMAXCONN statement, you must respecify the statement with the new parameters.

Examples

This example shows a SOMAXCONN statement specifying the default number of listening sockets.

```
SOMAXCONN                    1024
```

Usage notes

- A SOMAXCONN constant with a value of 10 is defined in the SOCKET.H header file. If your C socket programs use this constant to determine the acceptable maximum listening backlog queue length, remember to change the header file to specify the value that you specified for TCP/IP for the maximum_queue_depth on the SOMAXCONN statement.

SRCIP statement

Use the SRCIP statement to do the following tasks:

- Designate source IP addresses for certain outbound TCP connections or server applications

- Designate whether to prefer a public or a temporary IPv6 address when the algorithm for default source address selection is used to select the source IP address for certain outbound TCP connections or for outbound UDP or RAW packets

Restriction: Only one SRCIP block should appear in a configuration data set. Any subsequent SRCIP blocks are ignored and an informational message is displayed. If a syntax error is encountered when this statement is processed, an error message is displayed and the entire SRCIP block is ignored (no entries are processed).

Guidelines:

- The SRCIP block does not require you to specify the SOURCEVIPA parameter on either the IPCONFIG statement or the IPCONFIG6 statement.
- SRCIP JOBNAME and DESTINATION entries can appear in any order. If an outbound connection matches more than one JOBNAME or DESTINATION entry, the following order of precedence is used:
  1. A match on the most specific JOBNAME entry with at least one, non-wildcard character (this excludes JOBNAME *)
  2. A match on the most specific DESTINATION entry
  3. A JOBNAME * entry

Designating source IP addresses for TCP connections

The SRCIP statement supports a combination of JOBNAME and DESTINATION entries to designate source IP addresses. Use the SRCIP JOBNAME statement to designate source IP addresses to be used for TCP applications identified by specified jobs. Use the SRCIP DESTINATION statement to designate source IP addresses to be used for outbound TCP connections destined for specified IP addresses, networks or subnets.

These source IP addresses override source IP address specification based on:

- The VIPA IP addresses in the HOME list or the SOURCEVIPAINTERFACE specification, if SOURCEVIPA was specified
- The TCPSTACKSOURCEVIPA IP address

See the information about source IP address selection in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound connection is determined.

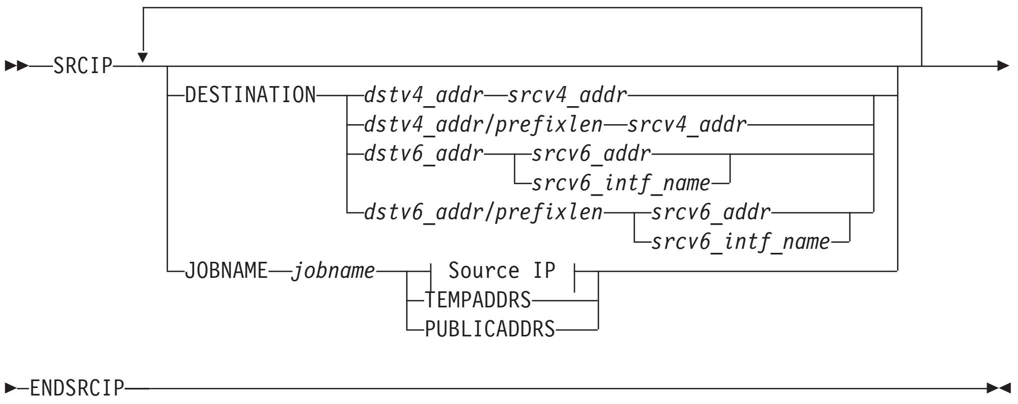
Guideline: Applications that bind to INADDR_ANY or to the unspecified IPv6 address (in6addr_any), and that match on a SRCIP JOBNAME or DESTINATION statement, do not have the designated IP address as its source address upon completion of the bind() call. The source address is not set to the designated address until completion of the subsequent connect() (client applications) or listen (server applications) call. This is important to note for applications that issue a getsockname() call after a bind() call.

Designating whether to prefer a public IPv6 address or a temporary IPv6 address

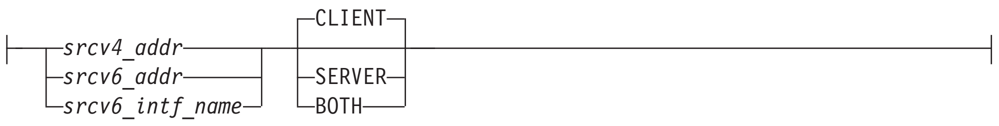
Use the SRCIP JOBNAME statement to indicate whether to prefer a public IPv6 address or a temporary IPv6 address for an application that is identified by the specified job name. The preference for temporary or public IPv6 addresses is part of selecting a source address for an outbound packet using the algorithm for source address selection. For more information about default source IP address selection and about using IPv6 temporary addresses, see z/OS Communications Server: IPv6 Network and Application Design Guide.

Syntax

Rule: Specify the parameters for JOBNAME or DESTINATION entries in the order shown here.



Source IP:



Parameters

DESTINATION
Designates a source address or interface to be used for outbound TCP connections with destinations that match the specified destination address or network.

Restriction: The source address specified in a matching SRCIP DESTINATION entry cannot be a distributed DVIPA unless the GLOBALCONFIG EXPLICITBINDPORTRANGE profile parameter is configured and one of the following situations exist:

- The application issued a connect request without a prior explicit bind.
- The source port for the outbound TCP connection socket was explicitly bound to port 0, to a specified port that is less than 1024, or to a port that is reserved for this job by a PORT or PORTRANGE profile statement.

If GLOBALCONFIG EXPLICITBINDPORTRANGE is not configured or if the source port is explicitly bound to an ephemeral port (equal to or greater than 1024) that is not reserved for this job, the connection request fails.

Rule: If the source port is less than 1024 or is a port that is reserved for this job and the specified source is a distributed DVIPA, you must ensure that multiple outbound connections to the same destination IP address and port cannot occur concurrently with the same source IP address and port.

A match to a SRCIP DESTINATION entry cannot be identified until a connect request is issued and the destination is known. However, some applications establishing outbound TCP connections might issue an explicit bind socket API prior to issuing the connect request. The port that is assigned or specified during this early bind processing might not be available across the sysplex for the destination-specific source IP address that is determined later at connect request time. If the port is not available, then an ambiguous situation might occur in which multiple outbound connections to the same destination IP address and port have the same source IP address and port. For this reason, the use of distributed DVIPAs on a SRCIP DESTINATION statement is not allowed without a GLOBALCONFIG EXPLICITBINDPORTRANGE statement configured on this stack.

Guideline: If you use distributed DVIPAs for the source IP address in a SRCIP DESTINATION entry, you should specify the SYSPLEXPORTS keyword on the VIPADISTRIBUTE statement for those distributed DVIPAs.

If duplicate destination values are specified in the SRCIP block, the first DESTINATION entry is used, any subsequent DESTINATION entry with a duplicate destination value is ignored, and a message is displayed.

If an outbound TCP connection's destination address matches more than one SRCIP destination address, the source address selected is determined by the most complete match. For example, suppose the following DESTINATION entries are specified in the SRCIP statement:

```
DESTINATION  9.67.0.0/16    10.1.1.1
DESTINATION  9.67.37.0/24   10.1.1.2
```

A destination address of 9.67.37.5 matches both DESTINATION entries, but 9.67.37.0/24 is the most specific match and 10.1.1.2 is selected as the source IP address for the outbound connection.

A DESTINATION designation is ignored if the job name for the connection matches any JOBNAME entry other than JOBNAME *.

dstv4_addr
: IPv4 host address to be matched by the destination IP address of an outbound TCP connection request. This is the destination IP address with which a specified source address is associated

dstv4_addr/prefixlen
: Subnet or network address to be matched by the destination IP address of an outbound TCP connection request. This is the destination IP subnet or network that a specified source address is associated with. The *dstv4_addr* value is a fully qualified IPv4 IP address and the *prefixlen* value is a decimal value in the range 1 - 32 that specifies how many of the leftmost contiguous bits of the address comprise the prefix.

dstv6_addr
: IPv6 host address to be matched by the destination IP address of an outbound TCP connection request. This is the destination IP address that a specified source address or interface name is associated with. See "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143 for a list of restrictions that must be observed when specifying this parameter. If the stack is not IPv6-enabled and an IPv6 IP address is specified, the DESTINATION entry is ignored and a message is displayed.

dstv6_addr/prefixlen
: Prefix address to be matched by the destination IP address of an outbound TCP connection request. This is the destination IP subnet or network that a

specified source address or interface name is associated with. The *dstv6_addr* value is a fully qualified IPv6 IP address and the *prefixlen* value is a decimal value in the range 1 - 128 that specifies how many of the leftmost contiguous bits of the address comprise the prefix.

See "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143 for a list of restrictions that must be observed when specifying this parameter. If the stack is not IPv6-enabled and an IPv6 IP address is specified, the DESTINATION entry is ignored and a message is displayed.

srcv4_addr
IPv4 host address to be used as a source IP address if the associated destination address is matched. The specified IP address does not need to be defined prior to the processing of the SRCIP block but it must be defined before the first TCP connect request is issued for the associated destination, otherwise the connect request fails.

The *srcv4_addr* value is a static VIPA, a dynamic VIPA, or a real IPv4 address associated with a physical interface. If a dynamic VIPA is used, it can be defined by a VIPADEFINE statement or previously activated with bind() or the IOCTL SIOCSVIPA value within a range of VIPAs.

Restrictions:
- An IPv4 source address cannot be specified for an IPv6 destination address.
- A distributed DVIPA cannot be specified for the source IP address in a DESTINATION entry unless the EXPLICITBINDPORTRANGE parameter on a GLOBALCONFIG statement is configured on this stack.

srcv6_addr
IPv6 host address to be used as a source IP address if the associated destination address is matched. The IPv6 IP address is in colon-hexadecimal format. A *prefix_length* cannot be specified. See "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143 for a list of restrictions that must be observed when specifying this parameter. If the stack is not IPv6-enabled and an IPv6 IP address is specified, the DESTINATION entry is ignored and a message is displayed. The specified IP address does not need to be defined prior to the processing of the SRCIP block, but it must be defined before the first TCP connect request is issued for the associated destination; otherwise, the connect request fails.

The *srcv6_addr* value is a static VIPA, a dynamic VIPA, or a real IPv4 address associated with a physical interface. If a dynamic VIPA is used, it can be defined by a VIPADEFINE statement or previously activated with bind() or a IOCTL SIOCSVIPA6 value within a VIPARANGE statement.

Restrictions:
- An IPv6 source address cannot be specified for an IPv4 destination address.
- A distributed DVIPA cannot be specified for the source IP address in a DESTINATION entry unless the EXPLICITBINDPORTRANGE parameter on a GLOBALCONFIG statement is configured on this stack.

srcv6_intf_name
The name of an IPv6 static VIPA, dynamic VIPA interface, or a physical interface to be used as a source interface if the associated destination address is matched. The maximum length is 16 characters. If a dynamic VIPA is used, it can be defined by a VIPADEFINE statement or previously

activated with bind() or the IOCTL SIOCSVIPA6 value within a VIPARANGE statement. If the stack is not IPv6-enabled, the DESTINATION entry is ignored and a message is displayed. The specified interface does not need to be defined prior to the processing of the SRCIP block, but it must be defined before the first TCP connect request is issued for the associated destination; otherwise, the connect request fails. If the interface has multiple IP addresses, then the source IP address for the outbound connection is selected from among these addresses according to the default source address selection algorithm. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.

Restrictions:

- An IPv6 source interface cannot be specified for an IPv4 destination address.
- A distributed DVIPA cannot be specified for the source IP address in a DESTINATION entry unless the EXPLICITBINDPORTRANGE parameter on a GLOBALCONFIG statement has been configured on this stack.

JOBNAME

Use this keyword to do one of the following:

- Designate a source address or interface to be used for TCP applications with a job name that matches the specified job name.
  - The parameters CLIENT, SERVER, and BOTH designate the type of socket function call on which the source IP address should be used.
  - The application can be a server if it binds to the IPv4 INADDR_ANY address or to the IPv6 unspecified address (in6addr_any), and the keyword SERVER or BOTH is specified with the SRCIP JOBNAME statement specified with a value other than JOBNAME *.

  See Designate source IP addresses for TCP connections for more information.
- Designate whether to prefer a temporary IPv6 address (TEMPADDRS) or a public IPv6 address (PUBLICADDRS) when the algorithm for default source address selection is used to select the source IP address for an application that has the specified job name.

  See Designate that a temporary IPv6 address is preferred over a public IPv6 address and Designate that a public IPv6 address is preferred over a temporary IPv6 address for more information.

If a connection request matches both a job name value other than JOBNAME * and a SRCIP destination address, the matching JOBNAME entry is used; otherwise, the matching DESTINATION entry is used.

Designate source IP addresses for TCP connections

Use the following parameters to designate source IP addresses for TCP connections.

jobname

Specifies the MVS job name of the application with which the specified source IP address is associated. The *jobname* value can be up to 8 characters in length. A trailing asterisk (*) indicates a wildcard specification. If you specify an asterisk (*), then all qualifying TCP applications, except those whose destination matches a SRCIP destination address on connect requests, are associated with the specified source IP address or interface; any existing specifications indicated by TCPSTACKSOURCEVIPA parameter are overridden. If similar prefixes are specified (for example, PAY* and PAYR*), then the actual source IP address associated with a job name is determined by the most complete match between the prefix and the job name. For example, an application whose job name is PAYROLL would match the PAYR* JOBNAME entry, not the PAY* JOBNAME entry.

If you want to associate one job name with both an IPv4 and an IPv6 IP address, specify two JOBNAME entries in which the *jobname* value is the same but the IP addresses are of different IP address families. If duplicate job names are specified in the same SRCIP block, and the duplicate entries specify an IP address of the same IP type (for example, both entries specify IPv4 or both specify IPv6 IP address types) the first JOBNAME entry is in effect. Any duplicate JOBNAME entries are ignored and messages are displayed.

If duplicate job names are specified in the same SRCIP block, and one of the entries specifies an IPv6 address or interface and the other entry specifies TEMPADDRS, the first JOBNAME entry is in effect. Any duplicate JOBNAME entries that specify an IPv6 address or interface or TEMPADDRS are ignored and messages are displayed.

Restriction: Unless a GLOBALCONFIG EXPLICITBINDPORTRANGE statement is configured on this stack, you cannot use an IPv4 SRCIP JOBNAME entry that specifies a distributed DVIPA to select as the source IP address for a connection on an IPv6 socket to an IPv4-mapped destination. If you do use such an entry, the connection fails.

srcv4_addr
IPv4 host address to be used as a source IP address if it matches the associated job name. The specified IP address does not need to be defined prior to processing the SRCIP block but it must be defined before the TCP connect or listen request is issued by the associated job; otherwise, the connect or listen request fails.

The *ipv4_address* value can be a static VIPA, a dynamic VIPA, or a real IPv6 address associated with a physical interface. If you specify a dynamic VIPA, it can be defined by a VIPADEFINE statement or a VIPARANGE statement. If the dynamic VIPA is defined by a VIPARANGE statement, then it must have been activated with a bind() or IOCTL SIOCSVIPA value.

srcv6_addr
IPv6 host address to be used as a source IP address if it matches the associated job name. The IPv6 IP address is in colon-hexadecimal format. You cannot specify a prefix length. See “Restrictions on IPv6 addresses configured in the TCP/IP profile” on page 143 for a list of restrictions that apply to this parameter. If the stack is not IPv6-enabled and an IPv6 IP address is specified, the JOBNAME entry is ignored and a message is displayed. The specified IP address does not need to be defined prior to processing the SRCIP block, but it must be defined before the TCP connect or listen request is issued by the associated job; otherwise, the connect or listen request fails.

The *ipv6_address* value is a static VIPA, a dynamic VIPA, or a real IPv6 address that is associated with a physical interface. If you specify a dynamic VIPA, it can be defined by a VIPADEFINE statement or a VIPARANGE statement. If the dynamic VIPA is defined by a VIPARANGE statement, then it must have been activated with a bind() or IOCTL SIOCSVIPA value.

srcv6_intf_name
The name of an IPv6 static VIPA, dynamic VIPA interface, or a physical interface to be used as a source interface if it matches the associated job

name. The maximum length is 16 characters. If a dynamic VIPA is specified, it can be defined by a VIPADEFINE statement or it could have been previously activated with bind() or IOCTL SIOCSVIPA6 value in a VIPARANGE statement. If the stack is not IPv6-enabled, the JOBNAME entry is ignored and a message is displayed. The specified interface does not need to be defined prior to processing the SRCIP block, but it must be defined before the TCP connect or listen request is issued by the associated job; otherwise, the connect or listen request fails. If the interface has multiple IP addresses, then the source IP address for the outbound connection is selected from among these addresses according to the default source address selection algorithm. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.

Guideline: When you are using a SRCIP JOBNAME statement for an IPv6 server application, code an IPv6 address (*srcv6_addr*) instead of an IPv6 interface (*srcv6_intf_name*); otherwise, the source address that is chosen for that IP interface might not be the best choice for the server application to be bound to. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.

CLIENT
Specifies that the source IP address should be used for client applications that are establishing outbound TCP connections that bind to the IPv4 INADDR_ANY address to IPv6 unspecified address (in6addr_any), or to the connect() call without having first completed a bind() call. The source IP address is determined on the subsequent connect() call. This value is the default.

SERVER
Specifies that the source IP address should be used to convert TCP server applications that bind to the IPv4 INADDR_ANY address or to the IPv6 unspecified address (in6addr_any), to bind to the specific source IP address. This means that only client applications that are using the designated IP address can connect to the server application. The source IP address is determined on the subsequent listen() call. When an application issues a getsockname() call after a bind() call to retrieve the source IP address, the processing is different from the processing that occurs when a TCP server application is converted to being bind specific using the BIND keyword on the PORT statements in the TCP/IP profile. When using the BIND keyword on the PORT statement, the designated IP address is set when the bind() call completes; some applications, such as DB2®, depend on this behavior.

BOTH
Specifies that the source IP address should be used for both client and server applications. For client applications, the source IP address is used for applications that invoke the bind() call with the IP INADDR_ANY address, the IPv6 unspecified address (in6addr_any), or the connect() call (for TCP outbound connections) without having first completed a bind() call. For server applications, the source IP address is used for applications that invoke the bind() call with the IP INADDR_ANY address or the IPv6 unspecified address (in6addr_any). The source IP address is determined on the connect() call for client applications and the listen() call for server applications.

Restriction: The options SERVER and BOTH are not valid with JOBNAME * specifications.

Designate that a temporary IPv6 address is preferred over a public IPv6 address

jobname
: Specifies the MVS job name of the application for which temporary IPv6 addresses should be preferred over public IPv6 addresses. The *jobname* value can be up to 8 characters in length. A trailing asterisk (*) is a wildcard specification.

  If duplicate job names are specified in the same SRCIP block and some of the duplicate entries specify an IPv6 address or interface and the other entries specify the value TEMPADDRS or PUBLICADDRS, the first JOBNAME entry is in effect. Any subsequent duplicate JOBNAME entries are ignored and messages are displayed.

TEMPADDRS
: Specifies that this JOBNAME entry causes temporary IPv6 addresses to be preferred over public IPv6 addresses for the specified job when default source IP address selection is performed for the outbound packets.

Designate that a public IPv6 address is preferred over a temporary IPv6 address

jobname
: Specifies the MVS job name of the application for which public IPv6 addresses should be preferred over temporary IPv6 addresses. The *jobname* value can be up to 8 characters in length. A trailing asterisk (*) is a wildcard specification.

  If duplicate job names are specified in the same SRCIP block and some of the duplicate entries specify an IPv6 address or interface and the other entries specify the value TEMPADDRS or PUBLICADDRS, the first JOBNAME entry is in effect. Any subsequent duplicate JOBNAME entries are ignored and messages are displayed.

PUBLICADDRS
: Specifies that this JOBNAME entry causes public IPv6 addresses to be preferred over temporary IPv6 addresses for the specified job when default source IP address selection is performed for the outbound packets. The application prefers public addresses by default.

Guideline: The environment in which the application is run determines the job name to be associated with a particular client or server application as follows:

- Applications submitted as batch jobs use the batch job name.
- The job name associated with applications started from the MVS operator console using the START command is determined as follows:
  - If the START command is issued with the name of a member in a cataloged procedure library (for example, S APP1), the job name is the member name (for example, APP1).
  - If the member name on the START command is qualified by a started task identifier (for example, S APP1.ABC), the job name is the started task identifier (for example, ABC). The started task identifier is not visible to all MVS components, but TCP/IP uses it to match a job name specified in the SRCIP block.
- The JOBNAME parameter can also be used on the START command to identify the job name (for example, S APP1,JOBNAME=XYZ).

- The JOBNAME parameter can also be included on the JOB card.
- Applications run from a TSO user ID use the TSO user ID as the job name.
- Applications run from the z/OS shell typically have a job name that is the user ID of the user that is logged on plus a 1-character suffix. Because this job name might not be predictable, you can use a job name ending in an asterisk (*) to ensure that these applications are governed by the SRCIP block. Because different applications might have the same job name, use care when you designate job names for applications running from the z/OS shell.
- Authorized users can run applications from the z/OS shell and use the _BPX_JOBNAME environment variable to set the job name. In this case, the value specified for the environment variable is the job name.
- z/OS UNIX applications started by INETD typically use the job name of the INETD server plus a 1-character suffix.

Steps for modifying

To modify parameters for the SRCIP block, consider the following:

- When a new SRCIP block is specified in a configuration data set on a VARY TCPIP,,OBEYFILE command, the new designations completely replace the existing designations.
- If you want to remove all the current designations, specify the SRCIP block without any entries, as indicated in the following example:
  ```
  SRCIP ENDSRCIP
  ```
- If you want to change one of the designations, first create a SRCIP block with the existing set of designations. Update any JOBNAME or DESTINATION entry with the designation that needs to be changed. Then issue the VARY TCPIP,,OBEYFILE command to activate the change.
- Changing the source IP address in a JOBNAME or DESTINATION entry affects only new TCP connect requests for the job or destination address. It does not affect processing for any existing connections.
- Changing the TEMPADDRS or PUBLICADDRS setting in a JOBNAME entry affects only new TCP connect requests; it does not affect processing for existing connections.

Guidelines:

- While the SRCIP-ENDSRCIP statement allows the specification of real IP addresses that are associated with physical interfaces, use static or dynamic VIPA interfaces. Because static and dynamic VIPA interfaces are not associated with a specific physical interface, they provide higher availability attributes in cases where specific network interfaces fail or where connectivity is lost in specific parts of the network. In cases where a real IP address must be specified as a source IP address on the SRCIP-ENDSRCIP block statement, there are several considerations that should be carefully evaluated:
  - The IP address specified affects only the source IP address that is used for all packets associated with an outbound TCP connection for the specified jobs or destinations; it does not influence the physical network interface selected by TCP/IP for any outbound packets associated with the TCP connection. TCP/IP determines the outbound interface by consulting its routing table and determining the best route to the destination IP address for the connection. As a result, the source IP address that is selected might not be associated with the outbound physical interface selected by TCP/IP. The network routing topology must allow for any inbound packets for this connection to be routed

  back to this TCP/IP host regardless of the network interface that was used for any outbound traffic associated with this connection.
  - If the physical network interface associated with a specified IP address fails or is deactivated, any incoming packets destined to this IP address might not be able to reach this TCP/IP host. This could disrupt traffic for both existing TCP connections and new TCP connections that use this source IP address.
- For JOBNAME entries, if the same VIPA source IP address is used on more than one z/OS TCP/IP stack, then the job-specific source IP address should be a distributed DVIPA with the SYSPLEXPORTS parameter enabled.

Tips: Give careful consideration if:
- The designated source is a dynamic VIPA.

  A dynamic VIPA that becomes inactive is no longer a valid designated source for SRCIP. A dynamic VIPA might become inactive if one of the following is true:
  - It is no longer a target for sysplex distribution on this stack
  - The application that causes its creation (in the case of an address created by a VIPARANGE statement) causes its deletion
  - It has been deactivated by the VARY TCPIP,,SYSPLEX,DEACTIVATE command
  - The DVIPA is in QUIESCING status
  - The TCP/IP stack leaves the sysplex group
- The designated source is an interface name.

  When an interface name is specified, it might be associated with multiple addresses. In this case, the address is chosen at connect time:
  - If an interface has multiple addresses defined, the address chosen as the source IP address for the outbound connection is selected according to the default source address selection algorithm. For more information, see the default source address selection algorithm information in z/OS Communications Server: IPv6 Network and Application Design Guide.
  - If the interface is a dynamic VIPA interface that is created by a VIPARANGE statement, then the actual address chosen as the source IP address for the outbound connection is not predictable or necessarily meaningful. Thus, you should specify an IPv6 address instead of an interface name if a VIPARANGE statement address is to be used.

Related topics
- "IPCONFIG statement" on page 190
- "IPCONFIG6 statement" on page 206

START statement

Use the START statement to start a device or interface that is currently stopped. This statement is usually specified at the end of *hlq*.PROFILE.TCPIP.

Requirements:
- VTAM must be active to START a device or interface with TCP/IP.
- Each device or interface to be started needs a separate START statement.

Tips:
- You can also use the VARY TCPIP,,START command to start a device or interface.

- The START statement can also be used in a VARY TCPIP,,OBEYFILE command data set to start the following:
  - A newly-defined device or interface
  - A device or interface stopped with the STOP statement
  - A device or interface that was never successfully started

Syntax

```
►►──START──┬─device_name────┬──────────────────────────────────────────────►◄
           └─interface_name─┘
```

Parameters

`device_name`
: The name of the device to start. This should be the same *device_name* specified on the DEVICE statement.

`interface_name`
: The name of the interface to start. This should be the same *interface_name* specified on the INTERFACE statement or the name of a dynamically created interface, such as an IQDX interface.

Steps for modifying

Modification is not applicable to this statement.

Examples

This example shows START statements that start devices LCS1 and LCS2.

```
START LCS1
START LCS2
```

Usage notes

- TCP/IP has a maximum of 255 non-VIPA started devices.
- There is no maximum number of static VIPA interfaces, but the maximum number of dynamic VIPA interfaces is 1024.
- The START statement is not valid for virtual devices or interfaces. When you use the DEVICE and LINK statements for IPv4, a virtual device is started automatically when a HOME entry is defined to it. When you use the IPv4 INTERFACE statement or when you use IPv6 , a virtual interface is started automatically when an INTERFACE statement is defined. The virtual device or interface never leaves the started (active) state.
- The START and STOP statements are processed after all other statements within the initial profile or VARY TCPIP,,OBEYFILE command data set.

Related topics

- “STOP statement”
- z/OS Communications Server: IP System Administrator's Commands

STOP statement

Use the STOP statement in a VARY TCPIP,,OBEYFILE command data set to stop a device or interface that is currently started.

Tip: You can also use the VARY TCPIP,,STOP command to stop a device or interface.

Syntax

```
►►──STOP──┬─device_name────┬──────────────────────────────────────────────►◄
          └─interface_name─┘
```

Parameters

device_name
: The name of the device to be stopped. This should be the same *device_name* specified on the DEVICE statement.

interface_name
: The name of the interface to be stopped. This should be the same *interface_name* specified on the INTERFACE statement or the name of a dynamically created interface.

Steps for modifying

Modification is not applicable to this statement.

Examples

This example shows STOP statements that stop devices LCS1 and LCS2.

```
STOP LCS1
STOP LCS2
```

Usage notes

- A virtual device or interface cannot be stopped.
- The START and STOP statements are processed after all other statements within the initial profile or VARY TCPIP,,OBEYFILE command data set.

Related topics

- "START statement" on page 292
- z/OS Communications Server: IP System Administrator's Commands

TCPCONFIG statement

Use the TCPCONFIG statement to update the TCP layer of TCP/IP.

Syntax

Tip: Specify the parameters for this statement in any order.

```
        ┌──────────────────────────────────────────────────┐
        ▼                                                  │
►►─TCPCONFIG─┬────────────────────────────────────────────┬┴──────────────────────────►◄
             ├─┬─CONNECTInitinterval 3000──────────────┬───┤
             │ └─CONNECTInitinterval milliseconds──────┘   │
             ├─┬─CONNECTTimeout 75───────┬─────────────────┤
             │ └─CONNECTTimeout seconds──┘                 │
             ├─┬─DELAYAcks───┬─────────────────────────────┤
             │ └─NODELAYAcks─┘                             │
             ├─┬─EPHEMERALPORTS 1024 65535──────────┬──────┤
             │ └─EPHEMERALPORTS  low_port high_port─┘      │
             ├─┬─FINWait2time 600───────────────┬──────────┤
             │ └─FINWait2time finwait2_seconds──┘          │
             ├─┬─FRRTHReshold 3────┬───────────────────────┤
             │ └─FRRTHReshold acks─┘                       │
             ├─┬─INTerval 120─────────────────────────┬────┤
             │ └─INTerval ──default_keepalive_interval┘    │
             ├─┬─KEEPALIVEPROBEInterval 75──────┬──────────┤
             │ └─KEEPALIVEPROBEInterval seconds─┘          │
             ├─┬─KEEPALIVEPROBES 10─────┬──────────────────┤
             │ └─KEEPALIVEPROBES number─┘                  │
             ├─┬─MAXImumretransmittime 120──────┬──────────┤
             │ └─MAXImumretransmittime seconds──┘          │
             ├─┬─NAGLE───┬─────────────────────────────────┤
             │ └─NONAGLE─┘                                 │
             ├─┬─QUEUEDrtt 20───────────┬──────────────────┤
             │ └─QUEUEDrtt milliseconds─┘                  │
             ├─┬─UNRESTRICTLowports─┬──────────────────────┤
             │ └─RESTRICTLowports───┘                      │
             ├─┬─RETRANSMITAttempts 15────┬────────────────┤
             │ └─RETRANSMITAttempts times─┘                │
             ├─┬─NOSELECTIVEACK─┬──────────────────────────┤
             │ └─SELECTIVEACK───┘                          │
             ├─┬─SENDGarbage ──FALSE─┬─────────────────────┤
             │ └─SENDGarbage ──TRUE──┘                     │
             ├─┬─TCPMAXRCVBufrsize 256K───────────────────────────┬┤
             │ └─TCPMAXRCVBufrsize ──tcp_max_receive_buffer_size──┘│
             ├─┬─TCPMAXSENDBufrsize 256K──────────────────────┬────┤
             │ └─TCPMAXSENDBufrsize tcp_max_send_buffer_size──┘    │
             ├─┬─TCPRCVBufrsize 65536─────────────────────┬────────┤
             │ └─TCPRCVBufrsize ── tcp_receive_buffer_size┘        │
             ├─┬─TCPSENDBfrsize 65536───────────────┬──────────────┤
             │ └─TCPSENDBfrsize ──tcp_send_buffer_size┘            │
             ├─┬─TCPTIMEstamp───┬──────────────────────────────────┤
             │ └─NOTCPTIMEstamp─┘                                  │
             ├─┬─TIMEWAITInterval 60──────┬────────────────────────┤
             │ └─TIMEWAITInterval seconds─┘                        │
             └─┬─NOTTLS─┬──────────────────────────────────────────┘
               └─TTLS───┘
```

Parameters

`CONNECTINITINTERVAL` *`milliseconds`*
The initial retransmission interval in milliseconds. The range is 100 - 3000. The default value is 3000.

`CONNECTTIMEOUT` *`seconds`*
The number of seconds before the initial connection times out. This connection includes TCP connections that are established over SMC-R links. The range is 5 - 190. The default value is 75.

`DELAYACKS | NODELAYACKS`

NODELAYACKS
Specifies that an acknowledgement is returned immediately when data is received that has the PUSH bit set on in the TCP header. Specifying the NODELAYACKS parameter on the TCPCONFIG statement overrides the specification of the DELAYACKS parameter on the TCP/IP stack PORT or PORTRANGE profile statements for the port used by a TCP connection, or on any of the following statements used to configure the route used by a TCP connection:

- The TCP/IP stack BEGINROUTES or GATEWAY profile statements
- The Policy Agent RouteTable statement
- The OMPROUTE configuration statements

DELAYACKS
Delays transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. This is the default, but the behavior can be overridden by specifying the NODELAYACKS parameter on the TCP/IP stack PORT or PORTRANGE profile statements for the port used by a TCP connection, or on any of the following statements used to configure the route used by a TCP connection:

- The TCP/IP stack BEGINROUTES or GATEWAY profile statements
- The Policy Agent RouteTable statement
- The OMPROUTE configuration statements

`EPHEMERALPORTS` *`low_port high_port`*
Indicates the range of ephemeral ports that are to be assigned at bind time. The default ephemeral port range is 1024 - 65535.

low_port
The starting port for the range of ports. The *low_port* value is in the range 1024 - 65535.

high_port
The ending port for the range of ports. The *high_port* value is in the range 1024 – 65535, and must be greater than or equal to the *low_port* value.

Guidelines:

- The TCP/IP stack selects an ephemeral port from the range of ports that are defined on the EPHEMERALPORTS parameter only if an ephemeral port was not assigned by EXPLICITBINDPORTRANGE, SYSPLEXPORTS or PASSIVEDATAPORTS processing.
- The SYSPLEXPORTS processing uses only ports that are within the range of ports that the EPHEMERALPORTS parameter defines.

- For the ports within the EPHEMERALPORTS range, if they are reserved by using port reservation definitions or the EXPLICITBINDRANGE parameter, they are excluded from the EPHEMERALPORTS port pool. Such exclusion effectively makes the pool smaller.

Restriction: Ports that are defined by BPXPARMS INADDRANYPORT and INADDRANYCOUNT must be restricted by the PORT or PORTRANGE statement to the job name OMVS. The stack does not assign these ports unless the user has the job name of OMVS.

FINWAIT2TIME *finwait2_seconds*
The number of seconds a TCP connection should remain in the FINWAIT2 state. The range is 1 - 3600. The default value is 600 seconds.

FRRTHRESHOLD *acks*
The threshold of duplicate ACKs for the functional recovery routine (FRR) to engage. The range is 1 - 2048. The default value is 3. Do not change this parameter from the default value unless a specific FRR-related problem occurs or you are under the direction of IBM service personnel.

INTERVAL *default_keepalive_interval*
The default TCP keepalive interval for applications that enable the SO_KEEPALIVE socket option and do not override the interval using the TCP_KEEPALIVE socket option. The range is 0 - 35791 minutes, and the default is 120. A value of 0 disables the keepalive function, so that sockets for which SO_KEEPALIVE is specified do not perform TCP keepalive. In this case, sockets specifying a specific interval using TCP_KEEPALIVE continue to send keepalive probes.

TCP keepalive probes end TCP connections after a period of inactivity. TCP keepalive is disabled by default for a connection, but can be enabled by issuing the SO_KEEPALIVE or TCP_KEEPALIVE socket options. The TCP_KEEPALIVE socket option enables the application to specify the keepalive probe interval, while the SO_KEEPALIVE socket option uses *default_keepalive_interval* as the interval.

After the interval has expired, TCP sends a single keepalive probe to the peer. If the TCP_KEEPALIVE socket option is not used to specify the probe interval, a total of ten probes are then sent at 75-second intervals if no response is received from the peer. If no response has been received 75 seconds after the last probe, the connection is reset. If TCP_KEEPALIVE is used to specify the keepalive probe interval, the number of probes and the interval between the probes might differ depending on the interval specified.

KEEPALIVEPROBEINTERVAL *seconds*
The interval in seconds between keepalive probes. The range is 1 - 75. The default value is 75.

This parameter does not change the initial keepalive timeout interval. It controls only the time between the probes that are sent out after the initial keepalive interval has expired.

KEEPALIVEPROBES *number*
The number of keepalive probes before the connection is aborted. The range is 1 - 10. The default value is 10.

This parameter does not change the initial keepalive time out interval. It controls only the number of probes that are sent out after the initial keepalive interval has expired.

MAXIMUMRETRANSMITTIME *seconds*

The maximum retransmit interval in seconds. The range is 0 - 999.990. The default value is 120.

Rule: If none of the following parameters is specified, this MAXIMUMRETRANSMITTIME parameter is used and the MINIMUMRETRANSMITTIME parameters of the following statements are not used:

- MAXIMUMRETRANSMITTIME on the BEGINROUTES statement
- MAXIMUMRETRANSMITTIME on the GATEWAY statement
- MAXIMUMRETRANSMITTIME on the ROUTETABLE statement
- Max_Xmit_Time on the OSPF_INTERFACE statement
- Max_Xmit_Time on the RIP_INTERFACE statement

The TCPCONFIG parameter value is used if no route parameter has been explicitly specified. If the TCPCONFIG parameter value of the maximum retransmit time is used, the MINIMUMRETRANSMITTIME value that is specified on the route parameter is not used, which means the minimum retransmit time is 0.

MAXIMUMRETRANSMITTIME of 0 indicates that the smallest retransmission interval must be used. When 0 is specified, TCP/IP uses a maximum retransmission interval of approximately 100 milliseconds. Specifying a very low maximum retransmission interval can result in additional system overhead for increased retransmission processing.

NAGLE | NONAGLE

NAGLE
Specifies that the Nagle algorithm is enabled. This is the default value.

NONAGLE
Specifies that the Nagle algorithm is disabled.

Rules:

- If the setsockopt() with TCP_NODELAY is specified for a connection, the Nagle algorithm is disabled for the connection.
- If NONAGLE is specified, the setsockopt() with TCP_NODELAY is ignored.

QUEUEDRTT *milliseconds*

The threshold at which the outbound serialization is engaged. The range is 0 - 50 milliseconds. The default value is 20 milliseconds. A value of 0 millisecond enables outbound serialization for all connections. Do not change this parameter from the default setting unless specific problems with outbound serialization occur or you are under the direction of IBM service personnel.

RESTRICTLOWPORTS | UNRESTRICTLOWPORTS

RESTRICTLOWPORTS

When set, ports 1- 1023 are reserved for users by the PORT and PORTRANGE statements. The RESTRICTLOWPORTS parameter is confirmed by the message:

```
EZZ0338I TCP PORTS 1 THRU 1023 ARE RESERVED
```

Restriction: When RESTRICTLOWPORTS is specified, an application cannot obtain a port in the 1- 1023 range unless it is authorized. Applications can be authorized to low ports in the following ways:

- Using PORT or PORTRANGE with the appropriate job name or a wildcard job name such as * or OMVS. If the SAF keyword is used on PORT or PORTRANGE, additional access restrictions can be imposed by a security product, such as RACF.
- APF authorized applications can access unreserved low ports.
- OMVS superuser (UID(0)) applications can access unreserved low ports.

Applications with a dependency on being able to obtain an available port in the 1- 1023 range without having that port explicitly reserved for its use should be run as APF authorized or superuser. Use RESTRICTLOWPORTS to increase system security.

UNRESTRICTLOWPORTS

When set, ports 1 - 1023 are not reserved. This is the default value. The UNRESTRICTLOWPORTS parameter is confirmed by the message:

```
EZZ0338I TCP PORTS 1 THRU 1023 ARE NOT RESERVED
```

RETRANSMITATTEMPTS *times*

The number of times that a segment is retransmitted before the connection is aborted. The range is 0 - 15. The default value is 15.

SELECTIVEACK | NOSELECTIVEACK

SELECTIVEACK

Enables the exchange of selective acknowledgements with partners that support the selective acknowledgement (SACK) option as defined by RFC 2018. For information about this RFC, see Appendix C, “Related protocol specifications,” on page 1465.

If TCP/IP initiates a TCP connection, then a selective acknowledgement permit option is sent. During a passive connect, if TCP/IP receives a TCP connection request with a selective acknowledgement permit option from a client and the SACK option is enabled, then TCP/IP sends a SYN-ACK with its own selective acknowledgement permit option. The SACK option must be enabled to help prevent unnecessary packets from being retransmitted when packet loss occurs in the network.

NOSELECTIVEACK

Disables the exchange of selective acknowledgements during connection setup and also during the entire connection. This is the default value.

SENDGARBAGE

Specifies whether the keepalive packets sent by TCP contain 1 byte of random data.

FALSE

Causes the packet to contain no data. This is the default value.

TRUE

Causes the packet to contain 1 byte of random data and an incorrect sequence number, assuring that the data is not accepted by the remote TCP.

TCPMAXRCVBUFRSIZE *tcp_max_receive_buffer_size*

The TCP maximum receive buffer size is the maximum value an application can set as its receive buffer size using SETSOCKOPT(). The minimum acceptable value is the value coded on TCPRCVBUFRSIZE, the maximum is 2M, and the default is 256K. If you do not have large bandwidth interfaces, you can use this parameter to limit the receive buffer size that an application can set.

Note: If Dynamic right sizing (DRS) is active for a connection, the TCPMAXRCVBUFRSIZE value is ignored and a maximum value of 2M is used. For more information about DRS, see TCP receive window in z/OS Communications Server: IP Configuration Guide.

IBM Health Checker for z/OS can be used to check whether the TCPMAXRCVBUFRSIZE value is sufficient to provide optimal support to the z/OS Communications Server FTP server. By default, it checks that TCPMAXRCVBUFRSIZE is at least 180 K. For more details about IBM Health Checker, see z/OS Communications Server: IP Diagnosis Guide.

TCPMAXSENDBUFRSIZE *tcp_max_send_buffer_size*
| The maximum send buffer size, which is between the value that is specified on
| the TCPSENDBUFRSIZE parameter and 2M. The default value is 256K.

TCPRCVBUFRSIZE *tcp_receive_buffer_size*
The TCP receive buffer size, which is between 256 bytes and the TCPMAXRCVBUFRSIZE value. The default value is 65536. This value is used as the default receive buffer size for those applications that do not explicitly set the buffer when they use SETSOCKOPT().

Increasing the receive buffer size does not allocate or consume any additional storage. The receive buffer size determines the amount of data that TCP/IP can buffer for the application to receive. When the TCP/IP stack receives the data, the data is stored in CSM data space or TCP/IP private storage. Each received segment has an associated data descriptor that resides in ECSA or TCP/IP private. No external mechanism controls which storage type is selected for the received data. For more information about the receiver buffer size and the TCP receive window, see TCP receive window in z/OS Communications Server: IP Configuration Guide.

TCPSENDBFRSIZE *tcp_send_buffer_size*
| The TCP send buffer size, which is between 256 bytes and the
| TCPMAXSENDBUFRSIZE value. The default value is 65536. This value is used as the default send buffer size for those applications that do not explicitly set the buffer size when they use SETSOCKOPT().

Increasing the send buffer size does not allocate or consume any additional storage. The send buffer size determines the amount of data that TCP/IP can buffer for the application to send. When the application sends the data, the TCP/IP stack stores the data in CSM data space. The sent data has one or more associated data descriptors that reside in ECSA.

TCPTIMESTAMP | NOTCPTIMESTAMP

NOTCPTIMESTAMP
TCP Timestamp Option is disabled, and MVS does not participate in TCP timestamp negotiation during connection setup and also during the entire life of connection.

TCPTIMESTAMP
TCP Timestamp Option is enabled. If MVS initiates a TCP connection, then a TCP timestamp option is sent. During a passive connect, for example, if MVS receives a TCP connection request with TCP timestamp option from a client and this option is enabled, then MVS sends a SYN-ACK with its own TCP timestamp option. This option should be enabled to help prevent wrapping of sequence numbers or to prevent a connection from receiving a delayed segment that was

originally intended for an earlier incarnation of the connection. The sequence numbers can wrap more quickly with higher bandwidth networks. This is the default value.

TIMEWAITINTERVAL *seconds*
The number of seconds that a connection remains in the TIMEWAIT state. The range is 0 - 120. The default value is 60.

TTLS | NOTTLS

NOTTLS
Indicates that the Application Transparent Transport Layer Security (AT-TLS) function is not activated for the TCP/IP stack. This is the default value.

TTLS Indicates that the AT-TLS function is activated for the TCP/IP stack. The AT-TLS function provides invocation of System SSL in the TCP transport layer of the stack. When a TCPCONFIG TTLS value is specified, the AT-TLS function uses AT-TLS policy information that is configured by using Policy Agent to determine how application connections are processed. If the setting is modified by using the VARY TCPIP,,OBEYFILE command, only new connections are affected by the change.

Guideline: If AT-TLS is enabled, you must activate the SERVAUTH class, define the INITSTACK resource profile, and permit users to it. For more information about AT-TLS data protection, see z/OS Communications Server: IP Configuration Guide.

Steps for modifying

To modify parameters for the TCPCONFIG statement, you must respecify the statement with the new parameters.

The parameter changes do *not* affect existing connections. They effect only new connections.

Examples

This example shows a TCPCONFIG statement that reserves ports 1 - 1023 for users by the PORT and PORTRANGE statements:

```
TCPCONFIG RESTRICTLOWPORTS
```

TRANSLATE statement

Use the TRANSLATE statement to indicate the relationship between an IP address and the physical address, on a specified link. You can use the TRANSLATE statement, with some limitations, for Ethernet, ATM, FDDI, and token-ring hosts that do not support ARP.

Restrictions:
- The TRANSLATE statement applies to IPv4 links only.
- The TRANSLATE statement is valid for LCS, ATM, and HCH devices only.

Each configuration data set's first TRANSLATE statement replaces the internal translation tables (the ARP table), including information dynamically added by ARP, with the new information. Subsequent TRANSLATE statements in the same data set add entries to the table.

If the first TRANSLATE statement of a profile contains no IP address or link name, all addresses are removed from the TRANSLATE list.

Syntax

Rule: Specify the parameters in the order shown here.

```
                 ┌───────────────────────────────────────────────────────────┐
                 ▼                                                           │
►►──TRANSLATE────internet_addr──┬─NSAP─────┬─physical_addr──link_name────┴────────►◄
                                ├─HCH──────┤
                                ├─ETHERNet─┤
                                ├─IBMTR────┤
                                └─FDDI─────┘
```

Parameters

internet_addr
: The IP address for which a translation is specified.

NSAP
: Indicates the network address is an ATM address.

HCH
: Indicates the network address is a HYPERchannel address.

ETHERNET
: Indicates the network address is an Ethernet address.

IBMTR
: Indicates the network address is a token-ring address.

FDDI
: Indicates the network address is an FDDI address.

physical_addr
: The network address corresponding to *internet_addr* and *link_name*. The format depends on the network type.

- For NSAP, specify a 40-digit hexadecimal value.

  Restriction: If the TRANSLATE statement is defining an ATM address for a TCP/IP stack on another z/OS system, then the last hex digit of the 40-digit address cannot be 0. (Zero is reserved for High Performance Routing (HPR) use by VTAM.)
- For HCH, specify a 12-digit hexadecimal number of the form *ttxxxxxxhhcc*.

  tt The trunk mask. Use values other than FF only when advised to do so by Network Systems Corporation or by a HYPERchannel expert.

  xxxxxx
  These 6 digits are ignored.

  hh The remote adapter address.

  cc The meaning depends on the type of remote adapter. If the remote adapter is attached to a VM TCP/IP or MVS TCP/IP system, then *cc* is the read port address (the lower of the two addresses that are attached to TCP/IP).

- For ETHERNET, IBMTR, and FDDI, specify a 12-digit hexadecimal MAC address of the remote adapter.
  - For Ethernet, the remote host is assumed to use network headers DIX Ethernet format, not the 802.3 format.
  - For token-ring, the translation table entry should not contain a token-ring source routed bridge path.

link_name
A network link name (from the LINK statement). The specified *internet_addr* is translated to the specified *net_addr* only when sending on this link. You can include multiple TRANSLATE statement entries for the same *internet_addr* with a different *link_name*.

Steps for modifying

To modify any values on the TRANSLATE statement, use a VARY TCPIP,,OBEYFILE command with a data set that contains a new TRANSLATE statement. All existing ARP entries are deleted. To remove all static ARP entries from the ARP table, specify an empty TRANSLATE statement.

Note:
1. If any HOME statement values were dynamically modified, all ARP static entries that correspond with the LINK names in the TRANSLATE statement are deleted and replaced.
2. If any DEVICE/LINK statement values were dynamically deleted, all static ARP entries that correspond with the LINK names on the TRANSLATE statement are deleted. Include a new TRANSLATE statement in the VARY TCPIP,,OBEYFILE command data set that contains the changed DEVICE/LINK statements and a new HOME statement.

For more information about the VARY TCPIP commands, see z/OS Communications Server: IP System Administrator's Commands .

Examples

This example shows the TRANSLATE statement for FDDI:

```
TRANSLATE
  9.67.51.3    FDDI    FF0000006702   FDDI1
  9.67.22.4    FDDI    FF0000009A05   FDDI1
```

Usage notes

- When using the TRANSLATE statement to define the ATM address of a TCP/IP stack on another z/OS system, it is important that the correct ATM address be specified. Start the ATM link on the other stack and use the Netstat ARP/-**R** ALL command to display the ARP cache. The entry for the local IP address in this display shows exactly which ATM address must be specified in the TRANSLATE statement on the first stack.
- Some token-ring hardware does not recognize the RFC 1469-mandated functional MAC address for multicast. The TRANSLATE statement can be used to configure a token-ring link to broadcast multicast datagrams as an alternative to using the functional MAC address. Use the reserved class D address 224.0.0.0 with one of the following special physical addresses:

– FFFFFFFFFFFF for all rings broadcast
– C00000040000 to reset back to the default functional address

The following list shows OSPF implementation examples of how to specify each method:

– All rings
  - TRANSLATE
  - 224.0.0.0 IBMTR FFFFFFFFFFFF linkname
– Assigned functional address:
  - TRANSLATE 224.0.0.0 IBMTR
  - C00000040000 linkname

The TRANSLATE statement is effective on a per link basis. You do not have to code a TRANSLATE statement if you want the assigned functional address, as it is the default method.

Related topics

- "DEVICE and LINK — ATM devices statement" on page 52
- "DEVICE and LINK — LAN Channel Station and OSA devices statement" on page 66
- "DEVICE and LINK — HYPERchannel A220 devices statement" on page 63

UDPCONFIG statement

Use the UDPCONFIG statement to update the UDP layer of TCP/IP.

Syntax

Tip: Specify the parameters for this statement in any order.

```
                  ┌──────────────────────────────────────────────────┐
                  ▼                                                  │
►►──UDPCONFIG─────┬──────────────────────────────────────────────────┬┴──────────►◄
                  │ ┌─EPHEMERALPORTS 1024 65535──────────┐            │
                  ├─┴─EPHEMERALPORTS  low_port  high_port─┴──────────┤
                  │ ┌─UNRESTRICTLowports─┐                            │
                  ├─┴─RESTRICTLowports───┴────────────────────────────┤
                  │ ┌─UDPCHKsum───┐                                   │
                  ├─┴─NOUDPChksum─┴───────────────────────────────────┤
                  │ ┌─UDPQueuelimit───┐                               │
                  ├─┴─NOUDPQueuelimit─┴───────────────────────────────┤
                  │ ┌─UDPRCVBufrsize 65535──────────────────────┐     │
                  ├─┴─UDPRCVBufrsize udp_receive_buffer_size────┴─────┤
                  │ ┌─UDPSENDBfrsize 65535───────────────────┐        │
                  └─┴─UDPSENDBfrsize udp_send_buffer_size────┴────────┘
```

Parameters

EPHEMERALPORTS *low_port high_port*

Indicates the range of ephemeral ports that are to be assigned at bind time. The default ephemeral port range is 1024 - 65535.

low_port

The starting port for the range of ports. The *low_port* value is in the range 1024 - 65535.

high_port

The ending port for the range of ports. The *high_port* value is in the range 1024 – 65535, and must be greater than or equal to the *low_port* value.

Guideline: Any ports that are reserved using port reservation definitions that are within the EPHEMERALPORTS range are excluded from the EPHEMERALPORTS pool, effectively making the pool smaller.

Restriction: Ports that are defined by BPXPARMS INADDRANYPORT and INADDRANYCOUNT must be restricted by the PORT or PORTRANGE statement to the job name OMVS. These ports will not be assigned by the stack unless the user has the job name of OMVS.

RESTRICTLOWPORTS | UNRESTRICTLOWPORTS

RESTRICTLOWPORTS

When set, ports 1 - 1023 are reserved for users by the PORT and PORTRANGE statements. The RESTRICTLOWPORTS parameter is confirmed by the message:

```
EZZ0338I UDP PORTS 1 THRU 1023 ARE RESERVED
```

Applications can be authorized to low ports in the following ways:

- By way of PORT or PORTRANGE with the appropriate job name or a wildcard job name such as * or OMVS. If the SAF keyword is used on PORT or PORTRANGE, additional access restrictions can be imposed by a security product (for example, RACF).
- APF authorized applications can access unreserved low ports.
- OMVS superuser (UID(0)) applications can access unreserved low ports.

Applications that have a dependency on being able to obtain an available port in the 1- 1023 range without having that port explicitly reserved for its use should be run as APF authorized or superuser. Use RESTRICTLOWPORTS to increase system security.

UNRESTRICTLOWPORTS

Ports 1 - 1023 are not reserved. This is the default value. The UNRESTRICTLOWPORTS parameter is confirmed by the message:

```
EZZ0338I UDP PORTS 1 THRU 1023 ARE NOT RESERVED
```

UDPCHKSUM | NOUDPCHKSUM

NOUDPCHKSUM

Used to ensure UDP does not do check summing. This option is ignored for UDP datagrams flowing over an IPv6 network, as UDP Checksum is a required function on an IPv6 network. If an AF_INET6 socket is used to send datagrams over an IPv4 network, this option disables the UDP checksum function.

UDPCHKSUM
Used to ensure UDP does check summing. This is the default value.

UDPQUEUELIMIT | NOUDPQUEUELIMIT

NOUDPQUEUELIMIT
Used to specify that UDP should not have a queue limit. With NOUDPQUEUELIMIT specified, it is possible for inbound datagrams to arrive and be queued to a UDP application's socket faster than the application can receive the datagrams. If so, the amount of data queued could be substantial, resulting in a possible shortage of system storage. For this reason, set a limit using UDPQUEUELIMIT or by using an IDS Traffic Regulation policy. The NOUDPQUEUELIMIT parameter is confirmed by the message:

```
EZZ0336I NO LIMIT ON INCOMING UDP DATAGRAM QUEUE SET
```

If intrusion detection services (IDS) Traffic Regulation (TR) policy is in effect for a UDP port then NOUDPQUEUELIMIT is overridden for that port.

UDPQUEUELIMIT
Used to set a queue limit for UDP. If set, then a maximum of 2000 incoming datagrams are queued on a UDP socket. This is the default value. The UDPQUEUELIMIT parameter is confirmed by the message:

```
EZZ0336I A LIMIT ON INCOMING UDP DATAGRAM QUEUE SET
```

If intrusion detection services (IDS) Traffic Regulation (TR) policy is in effect for a UDP port, the queue limit size is controlled by the policy for that port.

UDPRCVBUFRSIZE *udp_receive_buffer_size*
The UDP receive buffer size. Valid values are in the range 1 - 65535. The default is 65535.

UDPSENDBFRSIZE *udp_send_buffer_size*
The UDP send buffer size. Valid values are in the range 1 - 65535. The default is 65535.

Steps for modifying

To modify parameters for the UDPCONFIG statement, you must respecify the statement with the new parameters.

Examples

This example shows a UDPCONFIG statement that uses check summing, sets no queue limit, and sets the send buffer size to 8192:

```
UDPCONFIG UDPCHK NOUDPQ UDPSENDB 8192
```

VIPADYNAMIC statement summary

Use the VIPADYNAMIC statement to start a block of definitions related to dynamic VIPAs (DVIPAs) and the sysplex distributor; use an ENDVIPADYNAMIC statement to end the block of definitions. A VIPADYNAMIC block can contain the following statements:

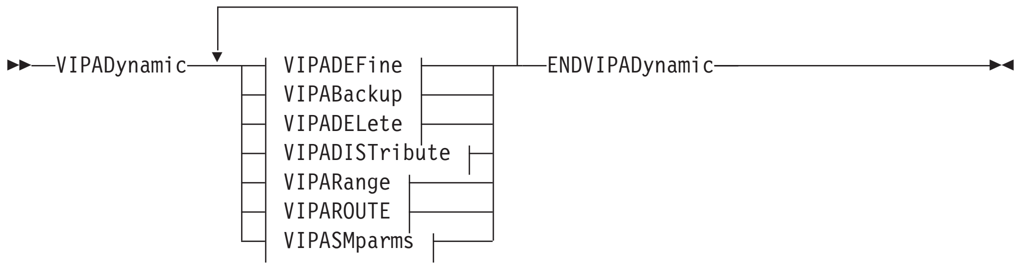
- “VIPADYNAMIC - VIPADEFINE statement” on page 308
- “VIPADYNAMIC - VIPABACKUP statement” on page 312

- “VIPADYNAMIC - VIPADELETE statement” on page 315
- “VIPADYNAMIC - VIPADISTRIBUTE statement” on page 316
- “VIPADYNAMIC - VIPARANGE statement” on page 335
- “VIPADYNAMIC - VIPAROUTE statement” on page 338
- “VIPADYNAMIC - VIPASMPARMS statement” on page 341

Rules:

- Within a single profile there should be only one VIPADEFINE or VIPABACKUP statement for a particular DVIPA. If the DVIPA does appear in more than one statement, a VIPADELETE statement must be specified before the last instance to ensure that it is not rejected.
- A stack is limited to no more than 1024 configured or target VIPAs at any one time. A configured dynamic VIPA is one that was created in any of the following ways, and might or might not be active:
  - Using VIPADEFINE
  - Using VIPABACKUP
  - Using an IOCTL SIOCSVIPA or SIOCSVIPA6 DEFINE value when this stack had a covering VIPARANGE statement
  - Using a BIND when this stack had a covering VIPARANGE statement
- Syntax errors in a VIPADYNAMIC block end further processing of the VIPADYNAMIC block. VIPADYNAMIC statements are processed up to the syntax error, and any remaining statements are ignored.
- The TCP/IP stack does not maintain interface counters for dynamic VIPA interfaces.

Syntax



Examples

This example shows the use of the VIPADEFINE, VIPADISTRIBUTE, VIPABACKUP, and VIPAROUTE statements within a VIPADYNAMIC/ ENDVIPADYNAMIC block.

```
VIPADYNAMIC
VIPADEFINE 255.255.255.192 201.2.10.11 201.2.10.12
VIPADISTRIBUTE DEFINE SYSPLEXPORTS TIMEDAFF 30 201.2.10.11
PORT 21 DESTIP 201.3.10.10 201.3.10.11
VIPABACKUP 100 201.2.10.13
VIPADEFINE DVIPA1 2001:0DB8:1::1
VIPADISTRIBUTE DISTMETHOD ROUNDROBIN DVIPA1 PORT 21 DESTIP ALL
VIPABACKUP 150 DVIPA2 2001:0DB8:2::2
VIPAROUTE 201.3.10.10  199.3.10.1
ENDVIPADYNAMIC
```

Related topics

- See z/OS Communications Server: IP Configuration Guide for more information about Virtual IP Addressing.
- "IPCONFIG statement" on page 190.
- "IPCONFIG6 statement" on page 206.

VIPADYNAMIC - VIPADEFINE statement

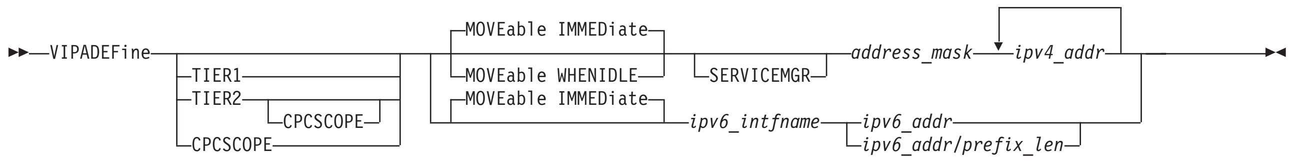
Designates one or more dynamic VIPAs (DVIPAs) that this stack should initially own and support. Other stacks can provide backup for these VIPAs if this stack fails.

Rule:

- Within a single profile there should be only one VIPADEFINE statement for a particular DVIPA. If the DVIPA does appear in more than one statement, you must specify a VIPADELETE statement before the last instance to ensure that the statement is not rejected.

Syntax

Rule: Specify the parameters in the order shown here.



Parameters

TIER1
: Indicates that the dynamic VIPA whose address is specified as an IP address on this statement are used to distribute incoming requests to z/OS or non-z/OS targets (for example, DataPower appliances).

  Restriction: You cannot configure this parameter on this statement if TIER2 is configured. A dynamic VIPA address cannot be used as both a TIER1 and TIER2 address.

TIER2
: Indicates that the dynamic VIPA whose address is specified as an IP address on this statement are used to distribute incoming requests from Tier 1 targets to the group of server applications that is named.

  Restriction: You cannot configure this parameter on this statement if TIER1 is configured. A dynamic VIPA address cannot be used as both a TIER1 and TIER2 address.

CPCSCOPE
: Indicates that the dynamic VIPA whose address is specified as an IP address on this statement is specific to the central processor complex (CPC) on which it is defined. That is, it is not moved to, or taken over, by another TCP/IP stack that is in a different CPC.

  Tier 1 non-z/OS target addresses must be on the same subnet as the subnet determined by the *address_mask* or *prefix_len* value.

Restrictions:

- A DVIPA defined with the CPCSCOPE parameter cannot be used in a VIPADISTRIBUTE DEFINE statement unless TIER2 is also configured .
- You cannot configure this parameter on this statement if TIER1 is configured.

MOVEABLE

This parameter is used to specify when a DVIPA that has been activated on this TCP/IP stack can be moved to another TCP/IP stack when the other TCP/IP stack requests ownership.

Rule: To preserve connections during dynamic VIPA takeover, you must specify the DYNAMICXCF parameter on the IPCONFIG statement for IPv4 DVIPA interfaces and on the IPCONFIG6 statement for IPv6 DVIPA interfaces.

IMMEDIATE
Specifies an immediate nondisruptive movement of a dynamic VIPA from one stack to another stack. This indicates that this dynamic VIPA can be moved to another stack as soon as the other stack requests ownership of the VIPA by executing a VIPADEFINE statement for the same dynamic VIPA. The new owning stack forwards packets for any existing connections to the original stack in order that the existing connections are not disturbed. All new connection requests are directed to the new owning stack. This is the default value.

The IMMEDIATE option is the only option supported for IPv6 addresses.

WHENIDLE
Indicates that this dynamic VIPA can be moved to another stack when there are no connections for this DVIPA on the current stack. While there are existing connections, any new connection requests continue to be directed to the current stack.

This option is not supported for IPv6.

SERVICEMGR
Indicates that sysplex distributor performs Multinode Load Balancing (MNLB) by functioning as a Service Manager (in place of Cisco's LocalDirector) for these distributed dynamic VIPAs. SERVICEMGR has no effect if a VIPADISTRIBUTE DEFINE statement does not exist for this VIPA.

Restrictions:

- MNLB is not supported for IPv6 DVIPAs; the SERVICEMGR parameter is ignored for these types of addresses.
- The SERVICEMGR parameter is ignored when TIER1, TIER2, or CPCSCOPE is specified.

address_mask
Specifies the subnet mask that determines how many of the bits of the IP address determine the subnet. All IP addresses in the same VIPADEFINE statement list must belong to the same subnet. That is, if the *address_mask* value is logically ANDed with all the IP addresses in the list, the resulting values must all be the same. The first IP address in the list determines the subnet.

When you specify the subnet mask for a DVIPA with a value of CPCSCOPE, ensure that the subnet is the same subnet that is used for tier 1 non-z/OS targets, such as DataPower appliances, that are routing requests to tier 2 targets on this CPC.

The *address_mask* value is specified in standard dotted decimal format; the IP addresses in the subnet must be a single contiguous range of IP addresses. A subnet mask of 0.0.0.0 is not valid.

Rules: The *address_mask* value must meet the following normal mask definition rules:

- When converted to binary, the most significant bit must be 1.
- When converted to binary, all bits less significant than (to the right of) the first 0 encountered must also be 0.

Restriction: This parameter applies only to IPv4 addresses.

`ipv4_addr`
Specifies the specific DVIPA to be defined. More than one *ipv4_addr* value can be specified on a single VIPADEFINE statement. A mixture of IPv4 addresses and an IPv6 interface on the same VIPADEFINE statement is not permitted. A mixture of VIPADEFINE statements with all IPv4 addresses and VIPADEFINE statements with IPv6 addresses is permitted within the same VIPADYNAMIC/ENDVIPADYNAMIC block, and the VIPADEFINE statements can be intermixed in any order.

If a DVIPA in this VIPADEFINE statement list is already active on another stack as a dynamic VIPA that was activated by VIPADEFINE or VIPABACKUP statement, the result of this VIPADEFINE statement depends on the level of each stack and how the DVIPA was originally defined.

If the DVIPA was originally defined with MOVE IMMEDIATE, then the original owning stack immediately gives up ownership of the DVIPA and the DVIPA is activated on this stack. If there were any connections to the DVIPA on the original owning stack, the newly owning stack forwards packets to the original stack in order that the existing connections are not disturbed.

If two or more stacks in the sysplex have the same DVIPA in VIPADYNAMIC VIPADEFINE statements, with different address masks, the stack that gets the active DVIPA determines the address mask.

If a DVIPA in this VIPADEFINE statement list is already active on this stack or another stack either as an IP address in a HOME statement or as a dynamic VIPA activated by way of an IOCTL or a BIND implicit activation, the DVIPA in the VIPADEFINE statement list is rejected and an error message is issued.

`ipv6_intfname`
The name of the interface. The maximum length is 16 characters. This specified name and the address specified in *ipv6_addr* are verified to ensure that the DVIPA interface is uniquely (consistently) defined throughout the sysplex environment.

`ipv6_addr`
Specifies the specific DVIPA to be defined. Only one *ipv6_addr* value can be specified on a single VIPADEFINE statement. A mixture of IPv4 addresses and IPv6 interfaces on the same VIPADEFINE statements is not permitted. A mixture of VIPADEFINE statements with all IPv4 addresses and VIPADEFINE statements with an IPv6 address is permitted within the same VIPADYNAMIC/ENDVIPADYNAMIC block, and the VIPADEFINE statements can be intermixed in any order.

If theDVIPA specified by the *ipv6_addr* value is already active on another stack as a dynamic VIPA that was activated by the VIPADEFINE or VIPABACKUP statement on the same interface name, the DVIPA is activated on this stack and changed to backup status on the other stack.

Requirement: All stacks (distributing stack, backup stack, target stack) which participate in distribution for a distributed DVIPA with IPv6 address must be at least z/OS V1R6.

If the specified *ipv6_addr* is already active on this stack or another stack either as an IP address on an INTERFACE statement or as a dynamic VIPA activated by way of an IOCTL or a BIND implicit activation, the VIPADEFINE statement is rejected and an error message is issued.

See "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143 for a description of the ipaddr_spec parameter and a list of restrictions that must be observed when specifying this parameter.

/prefix_len
Specifies the prefix length to be used when calculating a prefix for CPCSCOPE processing. The number of bits in the *ipv6_addr* value defines the prefix. Valid values are in the range 1 - 128.

When you specify the prefix length for a CPCSCOPE DVIPA, ensure that the prefix is the same subnet that is used for the tier 1 targets that are in this CPC.

Restriction: This parameter applies to IPv6 only.

Steps for modifying

- To remove one or more of the IPv4 addresses, use:
```
VIPADELETE  ipv4_addr  [ipv4_addr ...]
```
- To remove an IPv6 DVIPA interface, use:
```
VIPADELETE ipv6_intfname
```
  If the IPv4 DVIPA address or IPv6 DVIPA interface is being distributed, you must use one or more VIPADISTRIBUTE DELETE statements to end distribution before you can use the VIPADELETE statement to delete the DVIPA. The VIPADISTRIBUTE DELETE and VIPADELETE statements can appear in the same VARY TCPIP,,OBEYFILE command data set.
- To change the mask for one or more of the IPv4 addresses, you must first delete the IP addresses and then redefine them with the new mask:
```
VIPADELETE  ipv4_addr  [ipv4_addr ...]
VIPADEFINE  new_mask  ipv4_addr  [ipv4_addr ...]
```
  If the IP address is active, the VIPADELETE statement breaks any existing connections and causes the dynamic VIPA to be activated elsewhere in the sysplex if there is another stack prepared to activate it.
- To change the SERVICEMGR setting, you must first delete the DVIPA and then redefine it with the SERVICEMGR setting that you want.
- To change a VIPADEFINE from TIER1 to TIER2, from TIER2 to TIER1, from non-TIER to TIER, or from TIER to non-TIER, you must first specify a VIPADELETE *ipaddr* value.

Examples

```
VIPADEFINE  255.255.255.192 9.67.240.02
VIPADEFINE TIER2 CPCSCOPE V6DVIPA1 2000::9:67:240:2/96
```

VIPADYNAMIC - VIPABACKUP statement

Designates one or more dynamic VIPAs (DVIPAs) for which this stack provides automatic backup if the owning stack fails. Another stack is expected, but not required, to have this same DVIPA defined with a VIPADYNAMIC VIPADEFINE statement.

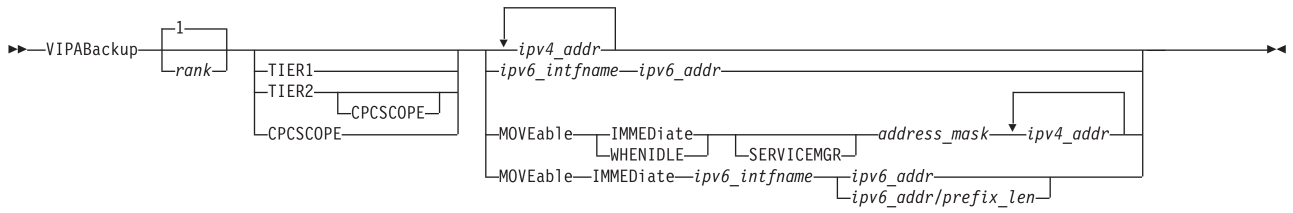
Rule:

- Within a single profile there should be only one VIPABACKUP statement for a particular DVIPA. If the DVIPA does appear in more than one statement, you must specify a VIPADELETE statement before the last instance to ensure that it is not rejected.

Syntax

Rule:

- Specify the parameters in the order shown here.



Parameters

rank
: Specifies the intended order of the VIPAs in this VIPABACKUP statement list in their respective backup chains, relative to other stacks in those backup chains. Larger numerical rank values move the respective stacks closer to the beginning of the backup chain.

  rank can be set to any integer from 1 (end of the backup chain) through 254 (start of the backup chain). Values 0 and 255 are reserved for use by the stacks themselves to temporarily force stack entries to the start or the end of the backup chain until an expected transition takes place.

  The default is a rank of 1.

TIER1
: Indicates that the dynamic VIPA whose address is specified as an IP address on this statement are used to distribute incoming requests to z/OS or non-z/OS targets (for example, DataPower appliances).

  Restriction: You cannot configure this parameter on this statement if CPCSCOPE is configured.

TIER2
: Indicates that the dynamic VIPA whose address is specified as an IP address on this statement are used to distribute incoming requests from Tier 1 targets to the group of server applications.

  Rule: If CPCSCOPE is also configured on this statement, then the Tier 2 group of server applications is limited to TCP/IP stacks on this CPC.

CPCSCOPE
Indicates that the dynamic VIPA whose address is specified as an IP address on this statement is specific to the central processor complex (CPC) on which it is defined. The VIPA is not moved to or taken over by another TCP/IP stack that is in a different CPC. A DVIPA defined with this characteristic can be used as the default route for incoming requests from Tier 1 targets on this CPC. A DVIPA defined with this characteristic can be used as the default route for incoming requests from non-z/OS tier 1 targets on this CPC. The non-z/OS Tier 1 target addresses must be on the same subnet as the subnet determined by the *address_mask* or *prefix_len* value.

Restrictions:
- A DVIPA defined with the CPCSCOPE parameter cannot be used in a VIPADISTRIBUTE DEFINE statement unless TIER2 is also configured.
- You cannot configure this parameter on this statement if TIER1 is configured.

MOVEABLE

This parameter is used to specify when a DVIPA that has been activated on this TCP/IP stack can be moved to another TCP/IP stack when the other TCP/IP stack requests ownership.

It can also be used to specify that the dynamic VIPA should be activated on this TCP/IP stack if it is not already active in the sysplex. If the DVIPA is already active in the sysplex when the VIPABACKUP statement is processed, this parameter is ignored. If you specify this parameter, you must also specify one of the following sets of dynamic VIPA information:
- The IPv4 address mask and address
- The IPv6 interface name and address

For more information about configuring VIPAs for activation with VIPABACKUP, see z/OS Communications Server: IP Configuration Guide.

IMMEDIATE
Specifies that the DVIPA can be activated immediately on another TCP/IP stack. The TCP connections to this TCP/IP stack are preserved. If the DVIPA has been activated on this TCP/IP, and the TCP/IP where the DVIPA is defined by a VIPADEFINE statement is subsequently activated, the DVIPA is activated immediately on that TCP/IP. And the TCP connections to this TCP/IP are preserved.

WHENIDLE
Specifies that the DVIPA remains active on this TCP/IP stack until there are no more connections to the DVIPA on this stack. If the DVIPA is activated on this TCP/IP, and the TCP/IP where the DVIPA is defined by a VIPADEFINE statement is subsequently activated, the DVIPA remains active on this TCP/IP until there are no more connections to the DVIPA on this TCP/IP.

This option is not supported for IPv6.

Guideline: Support for the WHENIDLE parameter is limited. It is recommended to use the IMMEDIATE parameter instead of the WHENIDLE parameter

SERVICEMGR
Indicates that sysplex distributor performs Multinode Load Balancing (MNLB) by functioning as a Service Manager (in place of Cisco's LocalDirector) for these distributed dynamic VIPAs. SERVICEMGR has no effect if a

VIPADISTRIBUTE DEFINE statement does not exist for the *dynamic* VIPA or VIPAs. SERVICEMGR is optional, and can be specified on a VIPABACKUP statement only when MOVEABLE is also specified.

This parameter is used only for activating the DVIPA when it is not already active in the sysplex. If the DVIPA is active when the VIPABACKUP statement is processed, this parameter is ignored.

Restrictions:

- MNLB is not supported for IPv6 DVIPAs; the SERVICEMGR parameter is ignored for these types of addresses.
- The SERVICEMGR parameter is ignored when TIER1, TIER2, or CPCSCOPE is specified.

address_mask

Specifies the subnet mask or prefix to be used when building the BSDROUTINGPARMS entry for this DVIPA when it is activated. This parameter can be specified on a VIPABACKUP statement only when MOVEABLE is also specified, and this parameter is required when MOVEABLE is specified on a VIPABACKUP statement. It is specified in standard dotted decimal notation. A subnet mask of 0.0.0.0 is not valid.

When you are specifying the subnet mask for a DVIPA with the value CPSCOPE, ensure that the subnet is the same subnet that is used for tier 1 non-z/OS targets that are being routed to tier 2 targets on this CPC.

This parameter is used only for activating the DVIPA when it is not already active in the sysplex. If the DVIPA is active when the VIPABACKUP statement is processed, this parameter is ignored.

Restriction: This parameter applies to IPv4 only.

ipv4_addr

Specifies the specific DVIPA to be backed up. More than one IPv4 address can be specified on a single VIPABACKUP statement. A mixture of IPv4 addresses and an IPv6 interface on the same VIPABACKUP statement is not permitted. A mixture of a VIPABACKUP statement with all IPv4 addresses, and a VIPABACKUP statement with an IPv6 interface, is permitted within the same VIPADYNAMIC/ENDVIPADYNAMIC block, and the VIPABACKUP statements can be intermixed in any order.

All *ipv4_addr* values specified on a single VIPABACKUP statement have the same rank. Use multiple VIPABACKUP statements to define different ranks for different *ipv4_addr* values.

The default LOOPBACK address (127.0.0.1) cannot be specified as the *ipv4_addr*.

ipv6_addr

Specifies the specific DVIPA to be backed up. Only one IPv6 address can be specified on a single VIPABACKUP statement. A mixture of VIPABACKUP statements with all IPv4 addresses, and VIPABACKUP statements with the IPv6 address, is permitted within the same VIPADYNAMIC/ENDVIPADYNAMIC block, and the VIPABACKUP statements can be intermixed in any order.

See "Restrictions on IPv6 addresses configured in the TCP/IP profile" on page 143 for a description of the *ipaddr_spec* parameter and a list of restrictions that must be observed when specifying this parameter.

/prefix_len
: Specifies the prefix length to be used when calculating a prefix for CPCSCOPE processing. The number of bits in the *ipv6_addr* value defines the prefix. The range is 1 - 128.

  When specifying the prefix length for a CPCSCOPE DVIPA, ensure that the prefix is the same subnet used for the tier 1 targets that are in this CPC.

  Restriction: This parameter applies to IPv6 only.

ipv6_intfname
: The name of the IPv6 interface to be backed up. The maximum length is 16 characters. Only one *ipv6_intfname* can be specified on a single VIPABACKUP statement. This specified name and the address specified in *ipv6_addr* are verified to ensure that the DVIPA interface is uniquely (consistently) defined throughout the sysplex environment.

Steps for modifying

- To remove an IPv4 address or IPv6 interface as a dynamic VIPA backup, use one of the following:
  - For an IPv4 address: VIPADELETE *ipv4_addr*
  - For an IPv6 interface: VIPADELETE *ipv6_intfname*
  - To change the rank (if the IP address is not currently active on this stack):
    ```
    VIPABACKUP  new_rank ipv4_addr
    ```
    However, if the IP address is currently active, you must first delete it and then configure it with the new rank by using one of the following:
    -
      ```
      VIPADELETE   ipv4_addr
      VIPABACKUP  new_rank  ipv4_addr
      ```
    -
      ```
      VIPABACKUP new_rank ipv6_intfname ipv6_addr
      ```
    or if the IP address is currently active, the VIPADELETE statement breaks any existing connections and causes the dynamic VIPA to activate elsewhere in the sysplex if there is another stack prepared to activate it.
- To remove an IPv6 interface and its address as a dynamic VIPA backup, use a VIPADELETE statement. *ipv6_intfname*.
- To modify the VIPABACKUP or VIPADELETE *ipv6_intfname* statement to remove an IP address as a dynamic VIPA backup, code the following:
  ```
  VIPADELETE ipaddr
  ```
- To change a VIPABACKUP from TIER1 to TIER2, from TIER2 to TIER1, from non-TIER to TIER, or from TIER to non-TIER, you must first issue a VIPADELETE *ipaddr*.

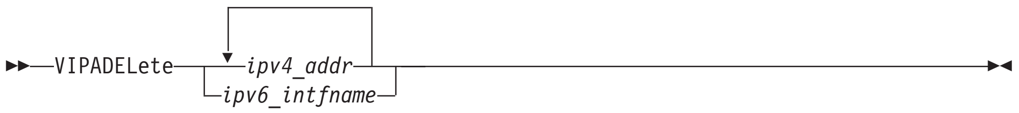
Examples

```
VIPABACKUP  V6DVIPA1 2000::9:67:240:2
VIPABACKUP  200 TIER2 CPCSCOPE MOVEABLE IMMEDIATE 255.255.255.192 9.67.240.02
```

VIPADYNAMIC - VIPADELETE statement

The VIPADELETE statement allows you to remove a dynamic VIPA interface from the VIPADEFINE or VIPABACKUP statement in which it occurs. It results in the interface and its dynamic VIPA being deleted.

Syntax



Parameters

ipv6_intfname
The name of the IPv6 interface as previously defined by a VIPADEFINE or VIPABACKUP statement. The maximum length is 16 characters. Only one interface name can be specified on a VIPADELETE statement.

ipv4_addr
Specifies the IPv4 IP address of the specific DVIPA to be deleted from the stack. More than one *ipv4_addr* value can be specified on a single VIPADELETE statement.

Examples

```
VIPADELETE 201.2.10.11
VIPADELETE DVIPA1
```

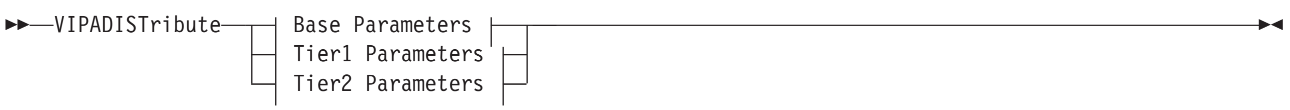
VIPADYNAMIC - VIPADISTRIBUTE statement

Enables (VIPADISTRIBUTE DEFINE) or disables (VIPADISTRIBUTE DELETE) the sysplex distributor function for a dynamic VIPA (defined on the same stack by a VIPADEFINE or VIPABACKUP statement) for which new connection requests can be distributed to other stacks in the sysplex. If you want to distribute FTP traffic, specify port 21 (or another designation according to which ports you are using for FTP) on the PORT parameter.

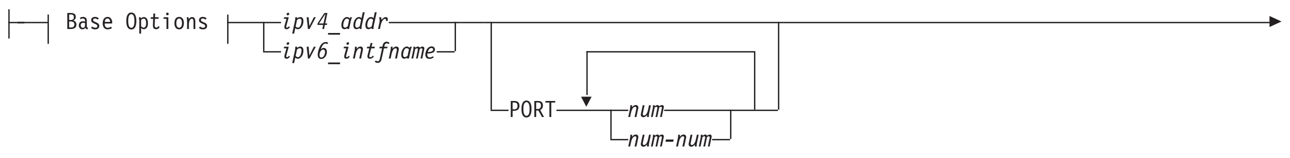
Tip: A target (or destination) DVIPA is one that was created on this stack as a result of a VIPADISTRIBUTE statement for an active VIPA on another stack. These addresses are identified by Flag I (internal only) in the Netstat HOME/**-h** command output.

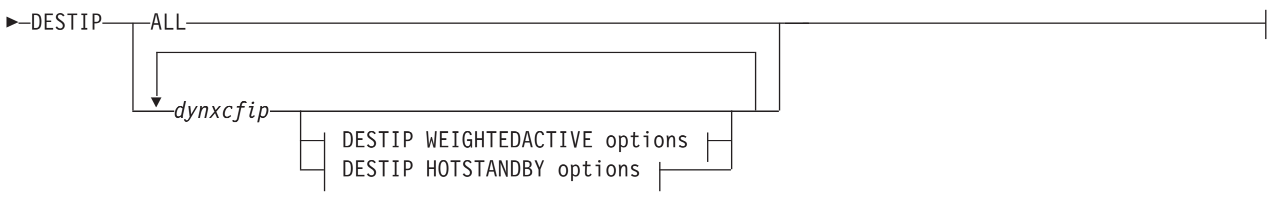
Syntax

Rule: Specify the parameters in the order shown here, except for the optional parameters preceding the IPv4 address or IPv6 interface name, which can be specified in any order.

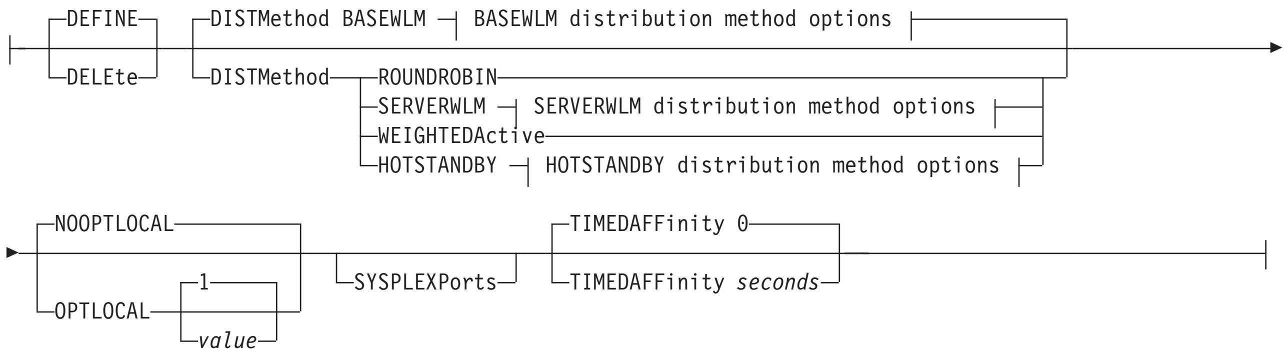


Base Parameters:

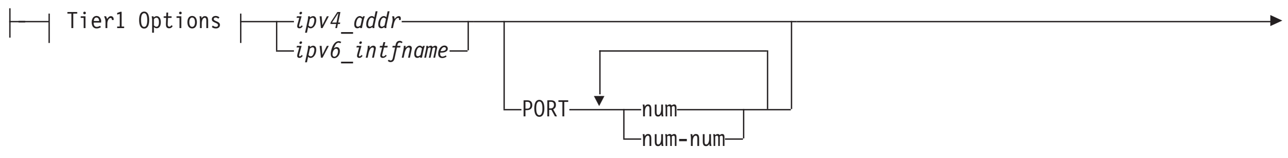


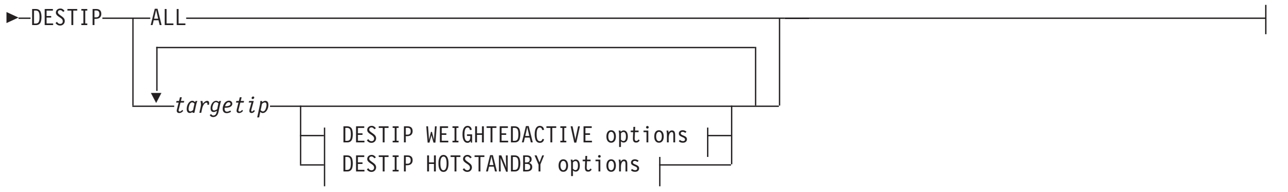


Base Options (These can be specified in any order):

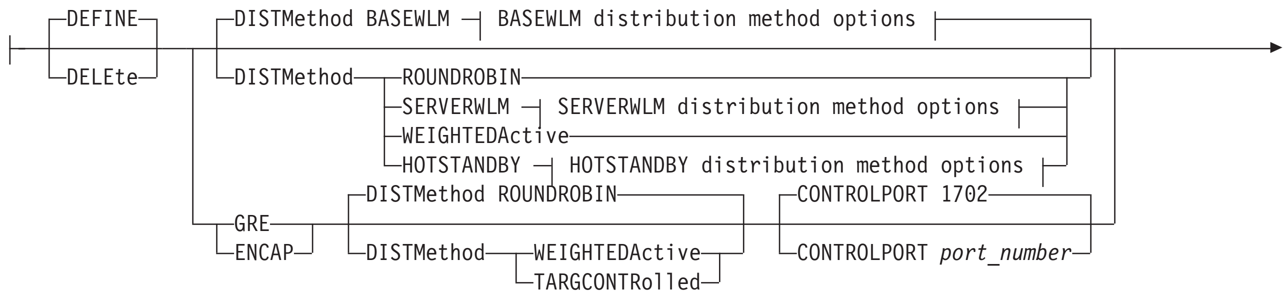


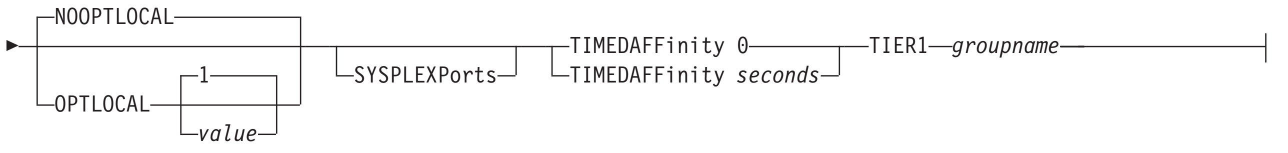
Tier1 Parameters:



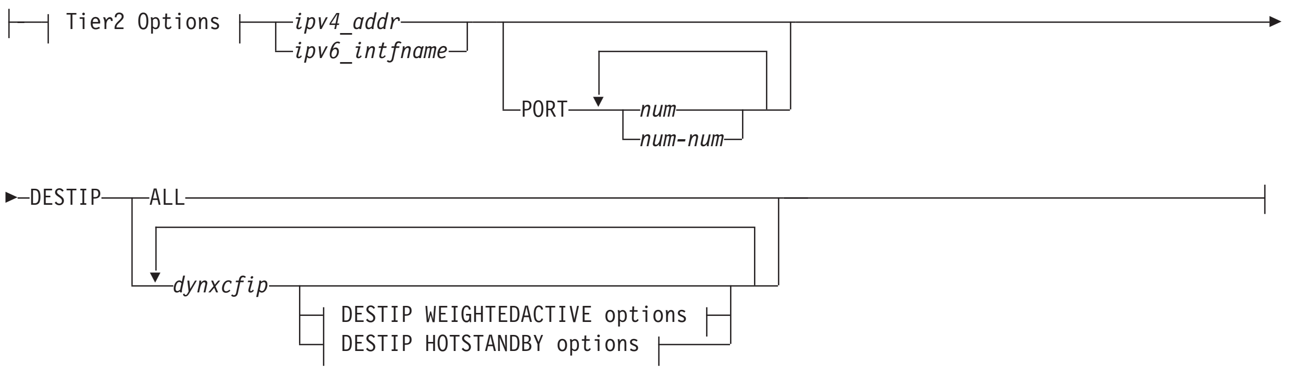


Tier1 Options (These can be specified in any order):

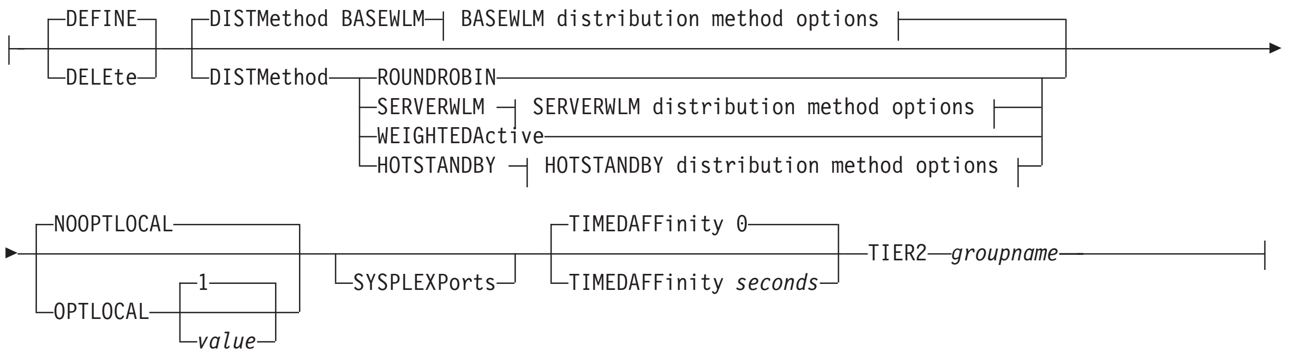




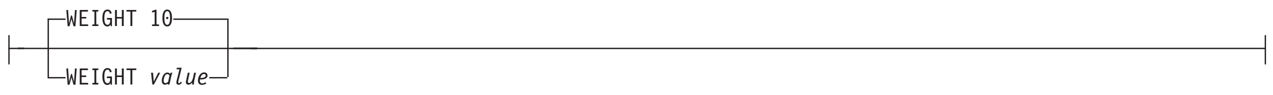
Tier2 Parameters:



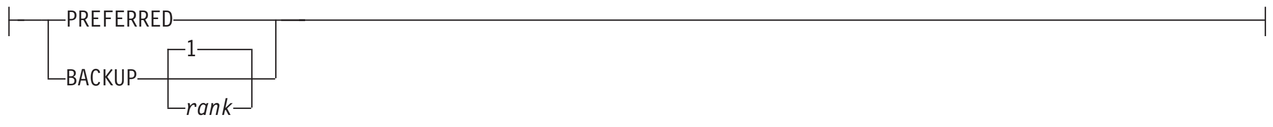
Tier2 Options (These can be specified in any order):



DESTIP WEIGHTEDACTIVE options:



DESTIP HOTSTANDBY options:



BASEWLM distribution method options:

```
   ┌─PROCTYPE  CP 1 ZAAP 0 ZIIP 0─┐
├──┼──────────────────────────────┼────────────────────────────────────────────────────────┤
   │           ┌──────────────────┐│
   │           ▼     ┌─CP 1─┐     ││
   └─PROCTYPE──────┬─┴─CP x─┴──┬──┴┘
                   │ ┌─ZAAP 0─┐│
                   ├─┴─ZAAP y─┴┤
                   │ ┌─ZIIP 0─┐│
                   └─┴─ZIIP z─┴┘
```

SERVERWLM distribution method options:

```
   ┌─PROCXCOST──ZAAP 1 ZIIP 1──────┐  ┌─ILWEIGHTING 0──────┐
├──┼───────────────────────────────┼──┼────────────────────┼───────────────────────────────┤
   │            ┌────────────────┐ │  └─ILWEIGHTING──┬─1─┬─┘
   │            ▼    ┌─ZAAP 1─┐  │ │                 ├─2─┤
   └─PROCXCOST─────┬─┴─ZAAP x─┴─┬┴─┘                 └─3─┘
                   │ ┌─ZIIP 1─┐ │
                   └─┴─ZIIP y─┴─┘
```

HOTSTANDBY distribution method options:

```
   ┌─AUTOSWITCHBACK───┐   ┌─HEALTHSWITCH───┐
├──┼──────────────────┼───┼────────────────┼────────────────────────────────────────────┤
   └─NOAUTOSWITCHBACK─┘   └─NOHEALTHSWITCH─┘
```

Parameters

DEFINE
: Adds or replaces the designation of this dynamic VIPA (defined on the same stack by a VIPADEFINE or VIPABACKUP statement) as distributable. This is the default value.

DELETE
: Deletes a previous designation of a dynamic VIPA as distributable.

DISTMETHOD
: Specifies the distribution method to be used by the distributing stack.

BASEWLM
: Specifies that Workload Manager (WLM) and policy information is used for this distributed DVIPA for incoming connection requests. Incoming connection requests are distributed according to relative WLM system weight preferences as modified by the Target Server Responsiveness (TSR) value, and possibly as modified by Service Policy Agent policies. The value DISTMETHOD BASEWLM is the default setting unless you specify GRE or ENCAP.

Restriction: You cannot specify DISTMETHOD BASEWLM if you specify GRE or ENCAP.

Rule: You must specify IPCONFIG SYSPLEXROUTING on all target systems to use this distribution method.

BASEWLM distribution method options:

PROCTYPE

This parameter is valid only when the distribution method is BASEWLM. zAAPs and zIIPs are specialty processors designed for specific application workloads. Some target applications can take advantage of these specialty processors. For workloads that use server-specific WLM weights, WLM typically returns a composite raw weight that takes into consideration how well the server is meeting its WLM goals with respect to the various types of processors the server is using. For workloads that use system-wide WLM recommendations, WLM is unaware of how a resource is utilizing the various processors. Instead, WLM returns a weight for each processor type that is based on the amount of displaceable capacity for this processor in the system as compared to the available capacity for this processor on the other target systems.

For applications that use specialty processors and receive WLM system weight recommendations, specify a PROCTYPE parameter to indicate the expected proportion of each type of processor that the target application's workloads should use. A composite recommendation is determined from these proportions. Each of the proportions should be expressed as a number in the range 0 - 99. Each proportion value is divided by the total to determine the processor usage pattern. To determine the processor proportions to configure, study your workload usage of assist processors by analyzing SMF records, using performance monitors reports, such as RMF™, and so on.

Possible values include:

CP *x*
The proportion of the workload that uses conventional processors.

ZAAP *y*
The proportion of the workload that uses zAAP processors.

ZIIP *z*
The proportion of the workload that uses zIIP processors.

For example, the value PROCTYPE CP 5 ZAAP 0 ZIIP 3 specifies a processor usage pattern such that 5/8 of the application's CPU utilization uses conventional processors (CP), and 3/8 of the application's CPU utilization uses zIIP processors.

For example, the value PROCTYPE CP 60 ZAAP 30 ZIIP 10, would specify a processor usage pattern such that 60% uses conventional processors (CP), 30% uses zAAP processors, and 10% uses zIIP processors.

The value PROCTYPE CP 1 ZAAP 0 ZIIP 0 is the default value; this value is used when the PROCTYPE parameter has never been specified. The default value indicates that 100% of the conventional processor weight (CP) should be considered when determining the composite weight (the application's workload does not use zIIP or zAAP processors). This value also disables an existing PROCTYPE value.

Specifying the PROCTYPE parameter without any parameters is equivalent to specifying the default values; you can use this setting disable an existing PROCTYPE value.

Restriction: When processor types are specified, at least one type must be specified with a nonzero value.

ROUNDROBIN

Specifies that WLM and policy information are not used to determine how to route future incoming connection requests for this distributed DVIPA. Incoming TCP connection requests are distributed in a round-robin fashion across the available TCP/IP stacks that are targets for each DVIPA/port combination and have at least one application server instance listening on the specified ports. This distribution method is not influenced by the number of server instances that are active on a target TCP/IP stack instance and listening on the same port (for example, SHAREPORT specified on the PORT reservation statement). In other words, a target TCP/IP stack that has multiple active servers on the same port does not receive more connection requests than a target stack that has a single instance of that server active. DISTMETHOD ROUNDROBIN is the default setting if GRE or ENCAP is specified.

Result: If a distribution target has a Target Server Responsiveness (TSR) value of 0, it is normally not used as a target for distribution. For more information about responsiveness monitoring, see z/OS Communications Server: IP Configuration Guide.

SERVERWLM

Specifies that server-specific WLM values should be collected for this group of DVIPA ports. If WLM server values can be collected for each target server, these values are used to distribute connections for this group of DVIPA ports [as modified by the Target Server Responsiveness (TSR) value, and possibly as modified by Service Policy Agent policies]. If all target servers do not provide the server-specific recommendations, then DISTMETHOD BASEWLM distribution is used instead. For more information about workload balancing and sysplex distribution, see z/OS Communications Server: IP Configuration Guide.

Rule: You must specify IPCONFIG SYSPLEXROUTING on all target systems to use this distribution method.

Result: zAAP and zIIP processor capacity is automatically included when SERVERWLM is specified and all systems in the sysplex are V1R9 or later.

Restriction: You cannot specify SERVERWLM if you specify GRE or ENCAP.

Port sharing

Specifying SHAREPORT on the PORT statement in the TCP/IP profile enables a group of servers to listen on the same port and thereby share the incoming workload. As new connections are received, the SHAREPORT algorithm distributes connections in a weighted round-robin fashion based on each server's Server accept Efficiency Fraction (SEF). By specifying SHAREPORTWLM on the PORT statement, connections are distributed in a weighted round-robin fashion based on the WLM server-specific recommendations, as modified by the Server accept Efficiency Fraction (SEF). If the shared port is a sysplex-distributed port and SERVERWLM is the distribution method that is being used, then SHAREPORTWLM should be coded on each target's PORT statement to take advantage of the new WLM server-specific recommendations when connections are received at the target; if it is not, new connections continue to be distributed using the existing SHAREPORT algorithm when they are received at the target.

Result: zAAP and zIIP processor capacity is automatically included when SHAREPORTWLM is specified and all systems in the sysplex are V1R9 or later.

SERVERWLM distribution method options:

ILWEIGHTING
: This parameter is valid only when the distribution method is SERVERWLM.

  The ILWEIGHTING parameter specifies the weighting factor that WLM uses when comparing displaceable capacity at different importance levels (ILs) as it determines a SERVERWLM recommendation for each system. The parameter value indicates how aggressively WLM should favor systems with displaceable capacity at low importance levels over systems with displaceable capacity at high importance levels. The higher the value specified for ILWEIGHTING the more a stack with displaceable capacity at lower importance levels is favored. See the internal load balancing information in z/OS Communications Server: IP Configuration Guide for more information about the effects of this parameter.

  0 WLM ignores importance levels when comparing displaceable capacity. This is the default value.

  1 WLM weights displaceable capacity that is at each successively lower importance level slightly higher than the capacity at the preceding importance level. The weighting increases proportionally to the square root of the difference between the two importance level values plus 1. This calculation provides a moderate bias when comparing displaceable capacity at different importance levels.

  Guideline: If you specify any value other than the default value (0), for the first time, specify this value (1) initially.

  2 WLM weights displaceable capacity that is at each successively lower importance level significantly higher than the capacity at the preceding importance level. The weighting increases proportionally to the difference between the two importance level values plus 1. This provides an aggressive bias when comparing displaceable capacity at different importance levels.

  3 WLM weights displaceable capacity that is at each successively lower importance level significantly higher than the capacity at the preceding importance level. The weighting increases proportionally to the square of the difference between the two importance level values plus 1. This provides an exceptionally aggressive bias when comparing displaceable capacity at different importance levels.

PROCXCOST
: This parameter is valid only when the distribution method is SERVERWLM.

  zAAPs and zIIPs are specialty processors designed to off-load specific application workloads. Some target applications are designed to have a portion of their workload take advantage of these processors.

  For server-specific recommendations, WLM calculates a composite weight based on a comparison, for each system, of the available capacity of each processor modified by the proportion of processor usage by the application. However, the composite weight does not consider that the conventional processor proportion on a system might be higher than normal because specialty processing capacity is constrained; a portion of the workload intended to run on a specialty processor ran on the conventional processor instead.

This parameter specifies a crossover cost which is applied to the zAAP or zIIP targeted workload that ran on the conventional processor; it reduces the conventional processor proportion which in turn reduces the composite weight for that system. This parameter can be used to cause WLM to favor systems that had less crossover (more of their workload running on the intended specialty processor) over systems that had more crossover. The higher the PROCXCOST crossover value, the more aggressively WLM recommendations favor systems with more specialty engine capacity which can reduce overall processing cost; however, if you use a PROCXCOST value that is too aggressive (high), overall workload performance for that service class might be sacrificed. The RMF Workload Activity Report shows the zAAP and zIIP processor utilization as well as how much crossover took place. Run this report before, and after, using the PROCXCOST parameter to better understand how this affects your overall workload performance.

Possible values include:

ZAAP *x*
The crossover cost of running targeted zAAP workload on a conventional processor instead of the zAAP processor, where *x* is an integer in the range 1 - 100. The higher the PROCXCOST zAAP value, the more aggressively the systems with less zAAP crossover occurring are favored. The default value is 1, which means that zAAP crossover is not considered.

ZIIP *y*
The crossover cost of running targeted zIIP workload on a conventional processor instead of the zIIP processor, where *y* is an integer in the range 1 - 100. The higher PROCXCOST zIIP value, the more aggressively the systems with less zIIP crossover are favored. The default value is 1, which means that zIIP crossover is not considered.

TARGCONTROLLED
Specifies that incoming connection requests are distributed using weights provided by the Tier 1 targets.

Restrictions:

- You can specify TARGCONTROLLED only when you specify GRE or ENCAP.
- You can specify TARGCONTROLLED only when you are distributing connections to DataPower appliances.

WEIGHTEDACTIVE
Specifies that WLM and policy information are not used to determine how to route future incoming connection requests for this distributed DVIPA. Instead, distribution of incoming TCP connection requests is balanced across the targets such that the number of active connections on each target is proportionally equivalent to a configured active connection weight for each target.

You can specify the weight for each target on the DESTIP parameter after each target's IP address. If you configure the value DESTIP ALL, then the default weight 10 is used and the connection distribution goal is to have an equal number of active connections for each DESTIP target. For more information, see the *DESTIP WEIGHTEDACTIVE options*.

This distribution method is not influenced by the number of server instances that are active on a target TCP/IP stack instance and listening on the same port (SHAREPORT parameter specified on the PORT reservation statement). For example, when two target TCP/IP stacks are configured with the same active connection weight, if one of the targets has multiple active servers for that port and the other target has only one instance of that server active, both stacks initially receive the same number of connection requests.

Rule: You must specify IPCONFIG SYSPLEXROUTING on all target systems to use this distribution method.

HOTSTANDBY

Specifies that there is at least one backup (hot-standby) target and a preferred target stack. You must configure one preferred target and at least one hot-standby target. For information about configuring these targets, see the PREFERRED and BACKUP parameters under *DESTIP HOTSTANDBY options*.

The target to which connections are distributed is referred to as the active target. If AUTOSWITCHBACK is configured, then the preferred target is the active target if it is available and has not had any health problems. If the active target becomes unavailable, the hot-standby target becomes the new active target, and the unavailable target becomes a hot-standby target.

A target is unavailable if any of the following conditions are true:

- The target is not ready.
- The distributor does not have an active route to the target.
- The target is not healthy; there is a severe problem detected by one of the following health metrics:
  - The target server responsiveness (TSR) value is 0%
  - WLM reported abnormal terminations are 1000 out of 1000 total transactions
  - WLM reported health is 0%

Requirement: PREFERRED or BACKUP must be configured on the DESTIP parameter for each target after the dynamic XCF address. For more information, see the *DESTIP HOTSTANDBY options*.

Rule: IPCONFIG SYSPLEXROUTING must be specified on all target systems for this distribution method to be used.

Restrictions:

- You cannot configure DESTIP ALL with this distribution method.
- You cannot configure GREor ENCAP with this distribution method.
- TIMEDAFFINITY is ignored with this distribution method.

Result: In the Netstat VDPT/-O report, ACTIVE is displayed if this is currently the active target, and BACKUP is displayed if this is currently a hot-standby target. For more information about the Netstat VDPT/-O report, see z/OS Communications Server: IP System Administrator's Commands.

HOTSTANDBY distribution method options:

AUTOSWITCHBACK | NOAUTOSWITCHBACK

AUTOSWITCHBACK
Specifies that the distributor automatically switches distribution back to the preferred target when the preferred target becomes available. For example, if the preferred target becomes a standby target because its server is no longer ready, when the server is again in the LISTENING state, the distributor automatically switches back to the preferred target as the active target. This is the default value.

Automatic switchback does not occur if the preferred target initially became unavailable because it was not healthy (TSR, WLM abnormal terminations, WLM health). A standby target that had health problems while active can look healthy again because it is not processing new work.

NOAUTOSWITCHBACK
Specifies that you do not want the distributor to switch back to the preferred target when it becomes available.

HEALTHSWITCH | NOHEALTHSWITCH

HEALTHSWITCH
Specifies that the distributor automatically switches from the active target if the target is not healthy. This is the default value.

NOHEALTHSWITCH
Specifies that the distributor ignores health metrics, and switches from the active target only if the target is not ready or if the distributor does not have an active route to the target.

SYSPLEXPORTS
Causes coordinated sysplex-wide ephemeral port assignment to be activated for the distributed DVIPA on all stacks where the DVIPA is defined, including all active candidate target stacks and the distributing stack, for all TCP connection requests.

SYSPLEXPORTS must be specified on the first VIPADISTRIBUTE statement for a DVIPA. It cannot be enabled after a DVIPA has been marked for distribution. If enabled, it cannot be disabled until all distribution has been deleted for the DVIPA (except for quiescing the DVIPA on the target stacks).

If you send connection requests to SYPLEXPORTS-enabled distributed DVIPAs and a random ephemeral port with no associated listener, then this connection times out.

Rules:

- For Passive Mode FTP to be distributed, the SYSPLEXPORTS parameter must be specified.
- Always specify the PORT parameter when specifying the SYSPLEXPORTS parameter; the way dynamic port allocation interacts with the EZBEPORT *vvtt* structure inhibits distribution to more than one target.
- A server application on a target stack can bind to a distributed DVIPA and a port that is not defined in the PORT parameter of the VIPADISTRIBUTE statement.
- If EPHEMERALPORTS is specified on the TCPCONFIG statement, only ports within the EPHEMERALPORTS range are assigned on the local stack for SYSPLEXPORTS processing.

Restrictions:

- For sysplexports allocation to function correctly, the stacks involved must be connected to the same sysplex ports coupling facility structure.
- SYSPLEXPORTS is ignored if GRE or ENCAP is specified.

TIER1 *groupname*

This parameter indicates that the dynamic VIPA whose address is specified as an IP address on this statement is used to distribute incoming requests to z/OS or non-z/OS targets (for example, DataPower appliances).

Rules:

- If you do not specify GRE or ENCAP, the targets are z/OS targets and the IP addresses specified on the DESTIP subparameter of this statement are dynamic XCF addresses.
- If you specify GRE or ENCAP, the targets are non-z/OS targets and the IP addresses specified on the DESTIP subparameter of this statement are not dynamic XCF addresses, but are the IP addresses of those Tier 1 targets.

The *groupname* value specifies the name of a cluster of equivalent server applications in the sysplex that the tier 1 targets might distribute the requests to. The groupname value can be 1 - 16 characters in length, must begin with an alphabetic character, and must not contain any national symbols, including @ or $. This value is used to correlate this statement with a corresponding TIER2 VIPADISTRIBUTE statement or statements. When TIER1 is specified, *groupname* is required, even if TIER2 definitions are not used.

CONTROLPORT *port_number*

Specifies the destination port number to be used when a control connection is being established to the Tier 1 target; a control connection is always established when GRE or ENCAP is configured. If CONTROLPORT is not specified, but GRE or ENCAP is specified, the default port number 1702 is used. See the DataPower configuration manuals for information about how to configure a control port.

Restrictions:

- You can specify CONTROLPORT only when you are distributing connections to non-z/OS targets.
- You can specify CONTROLPORT only when GRE or ENCAP is specified.
- The same port number (whether explicitly specified or specified by default) must be used on all VIPADISTRIBUTE statements that specify a CONTROLPORT value or that require a default port value. When a VIPADISTRIBUTE statement specifying a CONTROLPORT port_number (or requiring the default port value) has been encountered, any subsequent VIPADISTRIBUTE statement specifying a different port number is rejected.

GRE

Indicates that generic routing encapsulation (GRE) is used when distributing requests to the Tier 1 targets.

Restrictions:

- This parameter applies to IPv4 only.
- You can specify GRE only when you are distributing connections to non-z/OS targets.
- You can specify GRE only when TIER1 is specified.
- When GRE is specified, the following parameters on the VIPADISTRIBUTE statement are ignored:

  - SYSPLEXPORTS
  - OPTLOCAL
- When GRE is specified, the following DISTMETHOD parameters cannot be specified:
  - BASEWLM
  - SERVERWLM
- If the client is local to the distributing stack and it binds to inaddr_any or it uses an implicit bind, you must ensure that source IP address selection does not choose to use the distributed DVIPA destination as the source IP address for the connect() call. In the following example, source IP address selection will use the distributed DVIPA destination (DVIPA1) for the source because it is locally owned and advertised by the sysplex distributor. Because the SYN is routed by the sysplex distributor to a DataPower appliance, the packet is encapsulated in a GRE header:

GRE		**Inner IP header**	
Source IP	Dest IP	Source IP	Dest IP
XCF@	DataPower IP@	Distributed DVIPA1	Distributed DVIPA1

  After the GRE header is stripped away at the DataPower appliance, the connect() call will hang because the DataPower routing logic assumes that the packet should remain local because the source IP address (DVIPA1) is locally defined on the appliance as a non-advertised IP address.

  The source IP address must be routable by the DataPower appliance. You can use the SRCIP statement to select an appropriate source IP address for the client application.

  For more information about source IP selection, see z/OS Communications Server: IP Configuration Guide. For more information about the SRCIP statement, see “SRCIP statement” on page 282.

Tip: If you specify GRE, a control connection to the Tier 1 target is always established, using the CONTROLPORT port number as the destination port.

ENCAP

Indicates that routing encapsulation is used when distributing requests to the Tier 1 targets.

Restrictions:

- This parameter applies to IPv6 only.
- You can specify ENCAP only when you are distributing connections to non-z/OS targets.
- You can specify ENCAP only when TIER1 is specified.
- If you specify ENCAP, the following parameters on the VIPADISTRIBUTE statement are ignored:
  - SYSPLEXPORTS
  - OPTLOCAL
- If you specify ENCAP, you cannot specify the following DISTMETHOD parameters:
  - BASEWLM

    - SERVERWLM
- If the client is local to the distributing stack and it binds to the unspecified IPv6 address (in6addr_any), or if it uses an implicit bind, you must ensure that source IP address selection does not choose to use the distributed DVIPA destination as the source IP address for the connect() call. In the following example, source IP address selection will use the distributed DVIPA destination (DVIPA1) for the source because it is locally owned and advertised by the sysplex distributor. Because the SYN is routed by the sysplex distributor to a DataPower appliance, the packet is encapsulated in the outer IP header:

OuterIP header		**Inner IP header**	
Source IP	Dest IP	Source IP	Dest IP
XCF@	DataPower IP@	Distributed DVIPA1	Distributed DVIPA1

  After the outer IP header is stripped away at the DataPower appliance, the connect() call will hang because the DataPower routing logic assumes that the packet should remain local because the source IP address (DVIPA1) is locally defined on the appliance as a non-advertised IP address.

  The source IP address must be routable by the DataPower appliance. You can use the SRCIP statement to select an appropriate source IP address for the client application.

  For more information about source IP selection, see z/OS Communications Server: IP Configuration Guide. For more information about the SRCIP statement, see “SRCIP statement” on page 282.

Tip: If you specify ENCAP, a control connection to the Tier 1 target is always established, using the CONTROLPORT port number as the destination port.

TIER2 *groupname*
This parameter indicates that the dynamic VIPA whose address is specified as an IP address on this statement is used to distribute incoming requests from Tier 1 targets to the group of server applications that is named.

The *groupname* value specifies the name of a cluster of equivalent server applications in the sysplex that the Tier 1 targets might distribute the requests to. It is used to correlate this statement with a corresponding TIER1 VIPADISTRIBUTE statement.

The *groupname* value can be 1 - 16 characters in length, must begin with an alphabetic character, and must not contain any national symbols, including @ or $.

TIMEDAFFINITY *seconds*
Specifies whether or not a connection from a client (as identified by source IP address) to a particular server instance of several served by sysplex distributor shall establish an affinity for future connections from the same client (IP address) to the same Distributed DVIPA and ports. Valid values are in the range 0 to 9999. A value of 0, the default, means that no affinity is established when a new connection request is distributed to a particular server application instance by sysplex distributor. A nonzero value means that when a connection from a client is routed to a particular server instance, any subsequent connections from the same client (identified by source IP address) to the same

Distributed DVIPA and ports are routed to the same server instance until the specified number of seconds have elapsed after the last such connection was closed.

Restriction: Under some circumstances, a client's affinity with a specific target application server instance might be terminated prior to the specified time interval. This can occur if the key resources needed to satisfy new client TCP connection requests are not available. See z/OS Communications Server: IP Configuration Guide for more information.

If the TIMEDAFFINITY parameter is not initially specified on a VIPADISTRIBUTE statement, this indicates that timed affinity is not being used for the distributed DVIPA and ports, which is the same as specifying TIMEDAFFINITY 0.

Restriction: The TIMEDAFFINITY parameter cannot be specified with the OPTLOCAL keyword.

OPTLOCAL *value* **| NOOPTLOCAL**

NOOPTLOCAL
Causes target stacks to send locally originating connection requests to the sysplex distributor stack even when both endpoints reside on the same target stack. This is the default value.

OPTLOCAL *value*
Causes target stacks to optimize sysplex connections for which both endpoints reside on the same stack. When this value is specified, target stacks should bypass sending connection requests to the sysplex distributor stack for connections to a distributed DVIPA and port pair that reside locally, and instead process the connection locally using local optimizations. The local target stack continues to favor the local stack unless conditions on the local stack become unfavorable as defined by the value specified. If this happens, connections to this distributed DVIPA and port pair are sent to the sysplex distributor stack for appropriate work load balancing.

Restrictions:
- OPTLOCAL cannot be specified with the TIMEDAFFINITY keyword.
- OPTLOCAL is ignored if GRE or ENCAP is specified.

value
An integer in the range 0 - 16. The values 0 and 1 are special values, and values 2 - 16 are used as multipliers against the raw WLM weights.

A value of 0 indicates that connections originating from a target stack within the sysplex should always bypass sending the connection request to the sysplex distributor. The relative capacities of other target stacks within the sysplex are not considered in determining whether the connection should remain local.

A value of 1 indicates that connections originating from a target stack within the sysplex should always bypass sending the connection request to the sysplex distributor as long as the WLM weight for the server on the local stack is not 0. This is the default value if OPTLOCAL is specified without a value.

If a value in the range 2 - 16 is specified, the value is used as a multiplier against the local target stack's raw WLM weight to cause it to be favored over the other target stacks. The relative capacities of the other target stacks within the sysplex are considered in determining which stack should process the connection. The higher the value specified, the more the local stack is favored over other target stacks.

Regardless of the value specified on the OPTLOCAL parameter, if no local server is available, or the SEF is less than 75 or the abnormal transaction completions is greater than 250, or the health indicator is less than 75, connections are sent to the distributing stack.

Result: If the configured distribution method is ROUNDROBIN, WEIGHTEDACTIVE, or HOTSTANDBY, the OPTLOCAL value is forced to 0.

ipv4_addr
The specific IPv4 address for which the designation as distributable is to be deleted or defined.

Rule: An IPv4 address is not allowed if TIER1 ENCAP is specified.

ipv6_intfname
The specific IPv6 interface for which the designation as distributable is to be deleted or defined.

Rule: An IPv6 interface is not allowed if TIER1 GRE is specified.

PORT *num* **|** *num-num*
Specifies one or more individual ports, ranges of ports, or a combination of individual ports and ranges. Valid values for *num* are in the range of 1 - 65535. For a port range, the value for the second port must be greater than the first.

If the PORT parameter is specified, servers that bind to the specified DVIPA, the IPv4 INADDR_ANY address, or to the IPv6 unspecified address (in6addr_any) and one of the specified ports, cause the target stack to become eligible to receive connection requests.

The PORT parameter can also be omitted entirely from the VIPADISTRIBUTE statement. If the PORT parameter is omitted, then any server that binds a socket to the distributed DVIPA and a specific (nonzero) port, and establishes that socket as a listening socket, is eligible for connection workload balancing. The following methods can be used to bind a socket to the distributed DVIPA and a specific (nonzero) port:

1. If available, use a socket option provided by the server application to override the INADDR_ANY address and to specify a distributed DVIPA address for the listening port.
2. Code a BIND parameter that specifies a distributed DVIPA for the listening port in the TCP/IP profile PORT statement.
3. Use the TCP/IP profile SRCIP statement to specify a job name for the server application, the distributed DVIPA address, and the SERVER option. The listening port for the server application will be associated with the distributed DVIPA address.

Rules:

- When the PORT parameter is omitted from the VIPADISTRIBUTE statement and a specific (nonzero) port is not specified on the bind for the distributed DVIPA, then any ports that are bound to the distributed DVIPA are eligible for distribution.
- When the PORT parameter is specified, at least one port or port range must be specified. The maximum number of ports that is specified, including all individual ports and all ports within ranges, cannot exceed 64.

- Always specify the PORT parameter when specifying the SYSPLEXPORTS parameter. For the ports omitted from the PORT parameter, connection setup delays and connection timeouts might occur when there are no active listeners in the target stacks.
- For the ports specified in the PORT parameter for a distributed DVIPA, reserve them in the PORT statement in all target stacks associated with the distributed DVIPA so that ineligible server applications will not use them.

Requirement: If you specify TIER1 GRE or TIER1 ENCAP, you must specify the PORT parameter.

DESTIP *dynxcfip*

Specifies the dynamic XCF address (IPCONFIG DYNAMICXCF) of the TCP/IP stacks in the sysplex that are to be target stacks for the dynamic VIPA. The target stacks are candidates for receiving new incoming connection requests. See the PORT keyword for an explanation of how a candidate target stack becomes eligible to receive connection requests. If the VIPAROUTE statement specifies a target IP address for *dynxcfip*, but no route exists from the distributor to the target stack, that target stack is not considered for distribution, and the distributor treats this as it does when the dynamic XCF interface becomes inactive.

A maximum of 32 destination (target) dynamic XCF addresses can be specified.

Rules:
- If an IPv4 address is specified for this VIPADISTRIBUTE statement, then all of the addressees specified by the *dynxcfip* value must also be IPv4 addresses.
- If an IPv6 interface name is specified for this VIPADISTRIBUTE statement, then all of the addressees specified by the *dynxcfip* value must also be IPv6 addresses.

DESTIP *targetip*

When you specify TIER1 GRE or TIER1 ENCAP, this parameter specifies the IP address of non-z/OS hosts (for example, DataPower) appliances that are to be targets for the dynamic VIPA. The targets are candidates to receive new incoming connection requests.

When you specify TIER1 without specifying GRE or ENCAP, this parameter specifies the dynamic XCF address (IPCONFIG DYNAMICXCF) of the TCP/IP stacks in the sysplex that are to be target stacks for the dynamic VIPA. The target stacks are candidates to receive new incoming connection requests.

A maximum of 32 Tier 1 target IP addresses can be specified.

Requirement: You must specify TIER1 and either GRE or ENCAP when specifying the IP address of a non-z/OS host.

Rules:
- If an IPv4 address is specified for this VIPADISTRIBUTE statement, then all of the addressees specified by the *targetip* value must also be IPv4 addresses.
- If an IPv6 interface name is specified for this VIPADISTRIBUTE statement, then all of the addressees specified by the *targetip* value must also be IPv6 addresses.
- IPv6 addresses are not valid if TIER1 GRE is specified.
- IPv4 addresses are not valid if TIER1 ENCAP is specified.

DESTIP WEIGHTEDACTIVE options:

WEIGHT *value*

This parameter is configured following a DESTIP *targetip* or *dynxcfip* value.

This parameter has meaning only if the distribution method is WEIGHTEDACTIVE; it is ignored if this is not the distribution method. The weight is used by the distributor to determine the proportion of incoming requests to route to this target such that the number of active connections on each target is proportionally equivalent to the configured weight for each target. Valid values are in the range 1 - 99.

For example, if target 1 has a weight of 10 and target 2 has a weight of 90, then the connection distribution goal is to have 9 times as many active connections on target 2 as on target 1, or 10% of the active connections on target 1 and 90% of the active connections on target 2. If a weight is not specified, the default value of 10 is used. If the distribution method is WEIGHTEDACTIVE and weights are not configured for any targets, the goal is to have an equal number of active connections on each target.

Guidelines: Although weights can be in the range 1- 99, it is preferred to use weights that are greater or equal to 10. This is because the target server health metrics (Target Server Responsiveness [TSR] fractions) abnormal terminations, and the health indicator fractions are used to reduce the weight when these values are not optimal. By specifying weights greater than or equal to 10, these metrics can be applied without losing the original weight distinctions between targets. For example, if target 1 has a weight of 2, target 2 has a weight of 1, and a TSR for target 1 of 90% is applied, target 1 has a reduced weight of 1 (equal to target 2), but if target 1 has a weight of 20 and target 2 has a weight of 10, then when the TSR of 90% is applied to target 1, it has a weight of 18 (weight reduced, but it is still preferred over target 2).

If your workload has a low connection arrival rate (less than 100 connections per minute), and typically has a low number of active connections (less than 1000 active connections), you will get the most accurate distribution if you configure each weight so that it is a multiple of 10.

DESTIP HOTSTANDBY options:

PREFERRED

Specify this parameter after the dynamic XCF address (*dynxcfip*) on the DESTIP parameter. This parameter specifies that this address is the preferred target when the distribution method is HOTSTANDBY. If you configure AUTOSWITCHBACK, then the preferred target is the active target if it is available and has not had any health problems. If the active target becomes unavailable, the distributor switches to use a hot-standby target; the active target becomes a hot-standby target and the selected hot-standby target becomes the active target.

Restriction: You can specify this parameter only if you specify DISTMethod HOTSTANDBY.

BACKUP *rank*

Specify this parameter after the dynamic XCF address (*dynxcfip*) on the DESTIP parameter. This parameter specifies that this address is one of the backup targets when the distribution method is HOTSTANDBY.

The *rank* is used to determine which backup target is selected if the preferred target becomes unavailable. The backup with the highest rank is used. Valid values for *rank* are in the range 1 - 254; the default value is 1.

Restriction: You can specify this parameter only if you specify DISTMethod HOTSTANDBY.

DESTIP ALL
All TCP/IP stacks in the sysplex that have defined a dynamic XCF address of the same type as the IP address specified by the *ipv4_addr* or *ipv6_intfname* values in this VIPADISTRIBUTE statement are target stacks for the dynamic VIPA and for ports specified on this profile statement. If the distribution method WEIGHTEDACTIVE is being used, the default weight 10 is assumed for all targets; the goal is to have an equal number of active connections on each target.

Restrictions: .

-
  - DESTIP ALL cannot be specified when GRE or ENCAP is specified, or when the distribution method is HOTSTANDBY.
  - When DESTIP ALL or DESTIP dxcfaddr is specified, there is a limitation that only 32 targets can be used.

Steps for modifying

- To add ports (if the active VIPADISTRIBUTE statement has the PORT parameter coded) or destination stacks for a distributed DVIPA, use another VIPADISTRIBUTE statement to specify the additional port or ports and destination stacks. Ports and destination stacks for a distributed VIPA are cumulative, up to the maximum number allowed (64 for ports and 32 for destination stacks).
- To remove a port or a destination stack for IPv4, or both, for a distributed VIPA, use one of the following:

  `VIPADISTRIBUTE DELETE ipaddr PORT *port_num* ... DESTIP dynxcfip ...`

  `VIPADISTRIBUTE DELETE ipaddr PORT *port_num* DESTIP ALL`

  For IPv6, use one of the following:

  `VIPADISTRIBUTE DELETE *ipv6_intfname* PORT *port_num* ... DESTIP dynxcfip`

  `VIPADISTRIBUTE DELETE *ipv6_intfname* PORT *port_num* DESTIP ALL`
- To end distribution for a VIPA, use one or more VIPADISTRIBUTE DELETE statements to delete every port and destination stack that is currently configured for this VIPA. These changes are communicated to any stacks backing up the distribution of this DVIPA, unless the backup stack has its own VIPADISTRIBUTE statement coded.
- If ports are currently assigned for distribution dynamically for this Distributed DVIPA (PORT parameter omitted from the VIPADISTRIBUTE DEFINE), then VIPADISTRIBUTE DELETE can be used only to stop distribution for a target TCP/IP or for the Distributed DVIPA as a whole. VIPADISTRIBUTE DELETE cannot be used to stop distribution for a port with a Distributed DVIPA where ports are added dynamically.
- To specify certain ports for distribution when a distributed DVIPA is allowing distribution ports to be assigned dynamically (the active VIPADISTRIBUTE statement has no PORT parameter), you must first delete the VIPADISTRIBUTE statement (without PORT parameter). You can then code a VIPADISTRIBUTE statement with the PORT parameter. Existing connections to server instances are not affected. However, a server listening socket bound to a port which is not in the current PORT statement does not receive additional work.
- To allow dynamic port specification by having servers listening on ports when the active VIPADISTRIBUTE statement has a PORT parameter coded, you must first delete the VIPADISTRIBUTE statement (with PORT parameter). You can

then code a VIPADISTRIBUTE statement without the PORT parameter. Existing connections are not affected. Note that when the VIPADISTRIBUTE statement is specified without the PORT parameter, only servers that bind explicitly to the distributed DVIPA are eligible for workload distribution for that distributed DVIPA.

- To modify the OPTLOCAL option on the VIPADISTRIBUTE statement, respecify the VIPADISTRIBUTE statement with the new option value in a data set referenced by a VARY TCPIP,,OBEYFILE command. You can specify NOOPTLOCAL to dynamically stop OPTLOCAL processing.
- If the value TIMEDAFFINITY 0 is specified for a distributed DVIPA and ports for which a nonzero TIMEDAFFINITY value was in effect, no future affinities are established for new clients connecting to the distributed DVIPA and ports covered by the VIPADISTRIBUTE statement. Existing client affinities are not affected.
- If a nonzero TIMEDAFFINITY value is specified for an existing distributed DVIPA with active connections, affinities are established only for connections that are received at the distributing stack after the processing of the VARY TCPIP,,OBEYFILE command that established the nonzero TIMEDAFFINITY value. Existing connections do not automatically have an affinity established for the respective client.
- To change the distribution method being used, respecify the VIPADISTRIBUTE statement with the new DISTMETHOD option in a data set referenced by a VARY TCPIP,,OBEYFILE command. If the new distribution method is WEIGHTEDACTIVE:
  - Specify the WEIGHT keyword and the desired active connection weight value after each DESTIP dynamic XCF address.
  - If the active connection weight is not specified a default value of 10 is assumed.
  - If DESTIP ALL is specified, the active connection weight cannot be specified. A goal of having an equal number of active connections on all targets is used.
- To change the active connection weights being used for the targets when the distribution method is WEIGHTEDActive, in a data set referenced by a VARY TCPIP,,OBEYFILE command, respecify the VIPADISTRIBUTE statement with the WEIGHT keyword and the desired active connection weight value following each DESTIP dynamic XCF address. If the active connection weight is not specified, a default value of 10 is assumed.
- To change the PROCTYPE values being used with BASEWLM, in a data set referenced by a VARY TCPIP,,OBEYFILE command, respecify the VIPADISTRIBUTE statement with the PROCTYPE values for each processor type. If PROCTYPE is not specified, the previous values for PROCTYPE are used.
- To stop using PROCTYPE values, in a data set referenced by a VARY TCPIP,,OBEYFILE command, respecify the VIPADISTRIBUTE statement with the PROCTYPE values of CP 1 zAAP 0 zIIP 0, or simply PROCTYPE. This is the default usage of BASEWLM; only the general CPU weight that is returned by WLM is considered.
- To change the PROCXCOST values that are being used with SERVERWLM respecify the VIPADISTRIBUTE statement with the PROCXCOST values for each processor type in a data set referenced by a VARY TCPIP,,OBEYFILE command. If PROCXCOST is not specified, then the TCP/IP stack uses the previous values for PROCXCOST when it receives a server-specific WLM recommendation.
- To change the ILWEIGHTING value being used with SERVERWLM, respecify the VIPADISTRIBUTE statement with the ILWEIGHTING value in a data set

referenced by a VARY TCPIP,,OBEYFILE command. If ILWEIGHTING is not specified, then TCP/IP stack uses the previous values for ILWEIGHTING when it gets a server-specific WLM recommendation.

Examples

```
VIPADEFINE  255.255.255.192 9.67.240.02
VIPADISTRIBUTE
  DISTMETHOD SERVERWLM  9.67.240.02 PORT 10000  DESTIP ALL

VIPADEFINE TIER2 CPCSCOPE V6DVIPA1 2000::9:67:240:2/96
VIPADISTRIBUTE
  DISTMETHOD SERVERWLM PROCXCOST ZIIP 2 ZAAP 2 ILWEIGHTING 2
  TIER2 LOCALGROUP  OPTLOCAL 1  SYSPLEXPORTS
  V6DVIPA1 PORT 10000  DESTIP ALL
```

VIPADYNAMIC - VIPARANGE statement

Defines or deletes a subnet for which dynamic VIPA (DVIPA) activation requests, by way of a BIND, SIOCSVIPA IOCTL, or SIOCSVIPA6 IOCTL are honored. For guidance on defining this statement, see the APF-authorized application instance (ioctl) information and movement of unique application-instance (BIND) information in z/OS Communications Server: IP Configuration Guide.

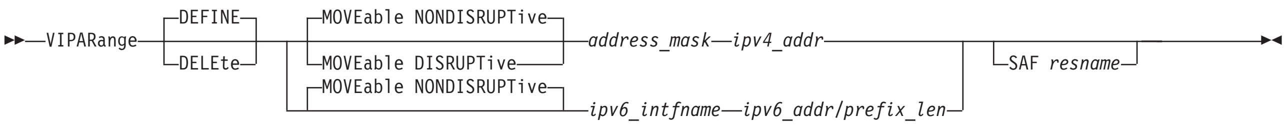
Guideline: VIPARANGE statements that are common to more than one stack should be defined in a common file and included in the appropriate stack profiles. This can help you avoid keying errors that could result in a failure to activate an application on a stack.

Rule: For any DVIPA creation request, the most specific VIPARANGE statement match (IP address and subnet) is used.

Restriction: There is a limit of 1024 VIPARANGE definition statements.

Syntax

Rule: Specify the parameters in the order shown here.



Parameters

DEFINE
: Specifies that this definition is to be added to the list of defined VIPARANGE definition statements. This is the default value.

DELETE
: Specifies that this definition (with the same *address_mask* and *ipv4_addr* values or the same *ipv6_intfname* and *ipv6_addr/prefix_len* values) is to be removed from the list of allowable ranges for IOCTL or BIND implicit dynamic VIPA activation.

  Tip: A VIPARANGE DELETE statement does not affect currently existing dynamic VIPAs in the range being deleted.

MOVEABLE NONDISRUPTIVE

Specifies an immediate nondisruptive movement of a dynamic VIPA from one stack to another stack. This value indicates that a dynamic VIPA in this VIPARANGE statement can be moved to another stack when that stack requests ownership of the DVIPA as the stack creates it; this occurs when an application binds to that DVIPA, the MODDVIPA utility is used to create the DVIPA through the SIOCSVIPA or SIOCSVIPA6 ioctl, or the application directly issues the SIOCSVIPA or SIOCSVIPA6 ioctl. The new owning stack forwards packets for any existing connections to the original stack in order that the existing connections are not disturbed. All new connection requests are directed to the new owning stack. The NONDISRUPTIVE option is the only option supported for IPv6 addresses and is the default value for IPv4 addresses.

MOVEABLE DISRUPTIVE

Indicates that nondisruptive movement does not occur for dynamic VIPAs created within this range on this stack. This option is not supported for IPv6.

A subsequent BIND on another stack for the same VIPA address fails. The VIPA on the original stack remains unchanged.

A subsequent SIOCSVIPA ioctl on another stack succeeds, and the VIPA on this stack is deleted. Any connections to the VIPA on this stack are broken.

address_mask

Provides the subnet mask that, when logically ANDed with the *ipv4_addr* value, determines the VIPARANGE subnet.

The address mask is specified in standard dotted decimal format for IP addresses. The *address_mask* variable is used only for IPv4. A subnet mask of 0.0.0.0 is not valid.

Rules: This value must meet the normal mask definition rules:

- When converted to binary, the most significant bit must be a 1.
- When converted to binary, all bits less significant than (to the right of) the first 0 encountered from the left must also be 0.

In other words, the IP addresses in the subnet must be a single contiguous range of IP addresses.

ipv4_addr

This determines a VIPARANGE subnet value when ANDed with the specified address mask. Any dynamic VIPA that is requested by way of IOCTL or by implicit BIND to a specific address must match a defined VIPARANGE subnet value, after the dynamic VIPA has been logically ANDed with the corresponding address mask.

ipv6_intfname

The interface name is used only for IPv6. This interface name is used for each DVIPA defined by this VIPARANGE statement.

ipv6_addr

This determines a VIPARANGE prefix defined by the *prefix_len* value.

Any dynamic VIPA that is requested by way of IOCTL or by implicit BIND to a specific address must match a defined VIPARANGE subnet value, after the dynamic VIPA has been logically ANDed with the corresponding network prefix.

/prefix_len
The number of bits in the *ipv6_addr* value defines the prefix. The range is 1 - 128.

SAF *resname*
Specifies the final qualifier of a System Authorization Facility (SAF) resource name. An application can create a dynamic VIPA in the specified VIPARANGE subnet if the user ID that is associated with the application is given READ access to the resource. The maximum length for the *resname* value is 8 characters.

For an application to create a dynamic VIPA in the VIPARANGE subnet, the user ID associated with the application must have access to the appropriate SAF resource:

- For an application that issues a bind socket call, the user ID must have READ access to the resource EZB.BINDDVIPARANGE.*sysname*.*tcpname*.*resname*.
- For an application that issues a SIOCSVIPA or SIOCSVIPA6 ioctl call or invokes the MODDVIPA utility (which issues the SIOCSVIPA or SIOCSVIPA6 ioctl call on behalf of the user), the user ID must have READ access to the resource EZB.MODDVIPA.*sysname*.*tcpname*.*resname*.

where:

- EZB.BINDDVIPARANGE and EZB.MODDVIPA are constant
- *sysname* is the value of the MVS &SYSNAME. system symbol
- *tcpname* is the name of the procedure used to start the TCP stack
- *resname* is the 1-8 character resource name that follows the SAF keyword on the VIPARANGE statement

Results:

- If the SAF keyword is specified and the user ID has READ access to the resource, the bind or ioctl call is processed.
- If the SAF keyword is specified and the user ID does not have READ access to the resource, the bind or ioctl call fails.
- If the SAF keyword is specified and the resource profile is not defined, the bind or ioctl call fails.
- Generic profiles are handled in the following ways:
  - All of the following profile specifications that include wildcard values match the EZB.BINDDVIPARANGE.*sysname*.*tcpname*.*resname* resource profile name:
    - EZB.BINDDVIPARANGE.*.*
    - EZB.BINDDVIPARANGE.**
    - EZB.BINDDVIPARANGE.*.*.*
  - All of the following profile specifications that include wildcard values match the EZB.MODDVIPA.*sysname*.*tcpname*.*resname* resource profile name:
    - EZB.MODDVIPA.*.*
    - EZB.MODDVIPA.**
    - EZB.MODDVIPA.*.*.*

  The most specific match to a resource profile is always used.

For more information about defining a security profile for SIOCSVIPA, SIOCSVIPA6, and MODDVIPA and about defining a security profile for binding to DVIPAs in the VIPARANGE statement, see z/OS Communications Server: IP Configuration Guide.

Steps for modifying

- To remove a VIPARANGE statement, use one of the following:

  `VIPARANGE DELETE` *`mask ipv4_addr`*

  `VIPARANGE DELETE` *`ipv6_intfname ipv6_addr/prefix_len`*

- To change the subnet for a VIPARANGE statement, use one of the following two methods:
  - To replace the subnet, use one of the following:

    ```
    VIPARANGE DELETE original_mask original_ipv4_addr
    VIPARANGE new_mask new_ipv4_addr
    ```

    ```
    VIPARANGE DELETE ipv6_intfname ipv6_addr/prefix_len
    VIPARANGE ipv6_intfname ipv6_addr/new_prefix_len
    ```

  - To enlarge the subnet, use one of the following:

    `VIPARANGE` *`mask2 ipv4_addr2`*

    `VIPARANGE` *`ipv6_intfname ipv6_addr/prefix_len2`*

  This configures a VIPARANGE statement where *mask2* ANDed with *ipv4_addr2* determines a subnet that overlaps or includes the original one.

  Alternatively, you can enlarge the subnet by using one of the following:

  `VIPARANGE` *`mask2 ipaddr2`*

  `VIPARANGE` *`ipv6_intfname ipv6_addr/prefix_len2`*

  This configures a VIPARANGE statement where *mask2* ANDed with *ipaddr* determines a subnet that overlaps or includes the original subnet.

Examples

```
VIPARANGE DEFINE 255.255.255.0 10.10.10.1
VIPARANGE DEFINE 255.255.255.255 10.10.10.210 SAF APPL1
VIPARANGE DEFINE 255.255.255.255 10.10.10.211 SAF APPL2
VIPARANGE 255.255.255.0 9.67.240.0
VIPARANGE V6DVIPARANGE 2000::9:67:270:0/112
```

VIPADYNAMIC - VIPAROUTE statement

A VIPAROUTE statement is used to select a route from a distributing stack or a backup distributing stack to a target stack. This route is used for distribution of all DVIPAs for which a matching dynamic XCF address, or ALL, was specified on a VIPADISTRIBUTE statement. This route is also used for forwarding packets to existing connections on a stack that contains the DVIPA in MOVING status. When processing a connection from the client, the sysplex distributor determines whether or not a matching VIPAROUTE statement has been specified. If it has, the best available route is determined using the normal IP routing tables. If no matching VIPAROUTE statement exists for that target, IP packets distributed by sysplex distributor to that target use dynamic XCF interfaces. Dynamic XCF interfaces include HiperSockets (iQDIO), IUTSAMEH for the same LPAR, or XCF interfaces created by the IPCONFIG DYNAMICXCF or IPCONFIG6 DYNAMICXCF statement. If the Cisco Multi-Node Load Balancing (MNLB) function is being used, the target IP address on a VIPAROUTE statement is used to route the packet directly to the target stack if a matching VIPAROUTE statement has been specified.

Rule:

- Ensure that the MTU value on the routes that are to be used is at least 604 (specify 1308 for IPv6). Lower MTU values can impact network performance and might result in loss of connections.

Result: There is always a matching route (and thus no message) if you define a default route by specifying DEFAULT.

If the VIPAROUTE statement specifies a target IP address for which no route exists, an informational message is issued the first time the problem is encountered. When this happens, that target is not considered for the distribution, and the distributor treats this the same way as when the dynamic XCF interface becomes inactive. If OMPROUTE is used for dynamic routing on the target, the GLOBALCONFIG SYSPLEXMONITOR DELAYJOIN TCP/IP profile option should be considered. The DELAYJOIN option delays the processing of sysplex-related definitions within the TCP/IP profile statements until OMPROUTE is active.

In the following cases, even though a VIPAROUTE statement has been specified, the dynamic XCF interface is used for distribution:
- A target IP address that is not owned by the target stack is specified
- The defined dynamic XCF address is for a pre-V1R7 target stack

Messages are issued at the distributing stack when these conditions are detected, and when the distributing stack first attempts to route a connection request to the target stack.

An additional case where the dynamic XCF interface is used even though the VIPAROUTE parameter has been specified is for a connection that is protected by an IPSec UDP-encapsulated security association negotiated with a peer behind a NAT.

Syntax

Rule: Specify the parameters in the order shown here.

```
                  ┌─DEFINE─┐
►►──VIPAROUTE─────┤        ├──dynxcfip──target_ipaddr────────────────────────────────►◄
                  └─DELEte─┘
```

Parameters

DEFINE
: Specifies that sysplex distributor should use the target IP address (*target_ipaddr*) to find the best available route to reach the target stack defined by the *dynxcfip* parameter. The target IP address can be any address in the HOME list of the target stack except for a dynamic VIPA (DVIPA) or a loopback address.

DELETE
: Specifies that a previously defined VIPAROUTE statement should be deleted. sysplex distributor processing for the target stack specified by the *dynxcfip* parameter reverts to using dynamic XCF interfaces for existing and new connections after approximately 60 seconds.

dynxcfip
: Specifies the IPv4 or IPv6 dynamic XCF address that uniquely identifies a

target stack. The address is defined with an IPCONFIG DYNAMICXCF or IPCONFIG6 DYNAMICXCF statement on that target stack.

If duplicate *dynxcfip* values are specified (with different *target_ipaddr* values) with the DEFINE function in the same profile, the first entry is in effect. Any duplicate entries are ignored and a message is displayed.

See “Restrictions on IPv6 addresses configured in the TCP/IP profile” on page 143 for a list of restrictions that must be observed when specifying this parameter for IPv6 dynamic XCF addresses.

target_ipaddr

Specifies any fully qualified IPv4 address (in dotted-decimal format) or fully qualified IPv6 address (in colon-hexadecimal format) in the HOME list of the target stack except for a dynamic VIPA (DVIPA) or a loopback address. The value is a static VIPA, a dynamic XCF address, or a real IPv4 or IPv6 address associated with a physical interface. See “Restrictions on IPv6 addresses configured in the TCP/IP profile” on page 143 for a list of restrictions that must be observed when specifying this parameter for IPv6 addresses. This IP address is used as a destination address for a target stack to obtain the best available route from the sysplex distributor to the target stack.

Specifying a static VIPA for this address might achieve the highest degree of fault tolerance. This alleviates the single point of failure issue with non-VIPAROUTE statement use of dynamic XCF interfaces. If an IP address is specified that is not owned by the target stack, dynamic XCF interfaces are used to distribute IP packets to this target stack.

For more information about the use of the routing information, see z/OS Communications Server: IP Configuration Guide.

Steps for modifying

- To remove the current configured statement, specify the VIPAROUTE DELETE statement with the same *dynxcfip* value and the same *target_ipaddr* value in a configuration data set referenced by a VARY TCPIP,,OBEYFILE command.
- To change the current configured statement, you must specify the VIPAROUTE DELETE statement with the same *dynxcfip* value and the same *target_ipaddr* value first, and then specify the VIPAROUTE DEFINE statement with the same *dynxcfip* value and the different *target_ipaddr* value referenced by a configuration data set on a VARY TCPIP,,OBEYFILE command.
- If the VIPAROUTE statement is changed, it affects active as well as new connections. For example, if an active connection is being distributed across dynamic XCF interface, and a VIPAROUTE DEFINE statement is defined for that target which results in the distributor selecting a route to the target over a different interface, then the active and new connections begin to use that new interface after approximately 60 seconds. If the previously defined VIPAROUTE statement is deleted, then active and new connections begin to use dynamic XCF interfaces after approximately 60 seconds.

Examples

This example shows how to define an alternate route to dynamic XCF for the sysplex distributor function.

```
VIPAROUTE 201.3.10.10 199.3.10.1
VIPAROUTE 2001:ODB8::201:3:10:10   2001:ODB8::199:3:10:1
```

VIPADYNAMIC - VIPASMPARMS statement

Defines service manager parameters. See z/OS Communications Server: IP Configuration Guide for more information about setting up sysplex distributor to be the Service Manager for Cisco's MNLB. This support applies only to IPv4.

Requirements:

- The VIPASMPARMS statement is required when any VIPADEFINE or VIPABACKUP statement in the profile contains the SERVICEMGR keyword, and it is permitted even if no active VIPADEFINE or VIPABACKUP statements in the profile currently contain the SERVICEMGR keyword.
- The VIPASMPARMS and SERVICEMGR parameters (on a VIPADEFINE statement) must be specified on the primary distributing stack (the stack identified by the VIPADEFINE statement) for the cluster address. The information is communicated to all backup stacks through expansion of the MVS XCF messaging messages used for normal DVIPA takeover processing. These parameters cannot be overridden on the backup stack.

Results:

- If the VIPASMPARMS statement is included without any VIPADEFINE or VIPABACKUP statements designated as SERVICEMGR, the values specified in this statement are saved and displayed in Netstat configuration displays.
- If a Distributed DVIPA is designated as SERVICEMGR on its VIPADEFINE statement, but one or both of SMMCAST group and SMPORT are not valid at the conclusion of profile processing, a console message is issued, and the Distributed DVIPA is not treated as SERVICEMGR at that point. However, the designation is saved in order that a subsequent VARY TCPIP,,OBEYFILE command that adds valid SMMCAST group and SMPORT values allows the cluster address to be treated as SERVICEMGR for all subsequent TCP connection requests.
- If a backup stack detects that inconsistent VIPASMPARMS statement values have been specified (by two different Sysplex stacks, for both of which this backup stack is backup for Distributed DVIPAs), this is considered a Sysplex configuration error, and the backup stack issues a console warning message.

Syntax

Rule: Specify the parameters in the order shown here.

```
►►──SMMCASTgroup ──ipaddr──SMPORT ──port──┬──────────────────────┬──────────────────────►◄
                                          └─SMPASSWORD ──password─┘
```

Parameters

SMMCASTGROUP *ip_addr*
: Specifies the multicast address used for communications between the sysplex distributor and the Cisco routers acting as forwarding agents.

SMPORT *num*
: Specifies the UDP port used for communications between the sysplex distributor and Cisco forwarding agents.

  The number is in the range 1 - 65 535.

SMPORT usage begins when the first dynamic VIPA with the service manager attribute is defined as distributable. At that point, a console message is issued if the same port value is already specified on an active PORT statement for that UDP port. Similarly, if a subsequent PORT statement is encountered after SMPORT usage begins for that same port, the subsequent PORT statement is rejected with a console message.

SMPASSWORD *string*

Specifies the password to enable MD5 encryption for all communication between sysplex distributor and forwarding agents. This is a 1 - 64 character alphanumeric string. For both the forwarding agents and the sysplex distributor, the password is treated simply as ASCII characters. No translation or conversion is performed.

Requirement: The password must match the one configured on Cisco forwarding agents.

Steps for modifying

To modify any of the parameters on this statement, you must respecify the statement with the changed parameter values.

Examples

```
VIPASMPARMS SMMCASTGROUP 224.0.0.10 SMPORT 50000
```

Chapter 3. TCP/IP cataloged procedure (TCPIPROC)

If you need to customize TCPIPROC, see the configuration information and customization information in z/OS Communications Server: IP Configuration Guide.

Copy the TCP/IP cataloged procedure in SEZAINST(TCPIPROC) to your system or recognized PROCLIB and modify it to suit your local conditions. Specify TCPIP parameters and remove or change the DD statements as required. The job name associated with the started task of the TCP/IP system address space must match the NAME parameter on the SUBFILESYSTYPE statement in the BPXPRMxx member of PARMLIB used to start z/OS UNIX.

Configuring the stack for IPv6 is done in BPXPRMxx. For more information about configuring the stack to support IPv6, see z/OS Communications Server: IP Configuration Guide or z/OS Communications Server: IPv6 Network and Application Design Guide.

The TCP/IP cataloged procedure is used to specify parameters and define input/output files to be used by the stack. One of the main input data sets defined in the cataloged procedure is the Profile data set. This data set is defined by the PROFILE DD statement.

Specifying TCP/IP address space parameters

Parameters are specified in the PARM= field of the cataloged procedure's EXEC JCL statement. The values specified in this field can be any of the following ones:

Stack initial component trace parameters

The following parameters configure stack tracing at initialization time:

- CTRACE(CTIEZBxx) or TRC=xx can be specified to identify the CTIEZBxx member of SYS1.PARMLIB, which contains the SYSTCPIP Component Trace (CTRACE) options. If neither parameter is specified, the default member CTIEZB00 is used.
- IDS=xx can be specified to identify the CTIIDSxx member of SYS1.PARMLIB, which contains the SYSTCPIS Component Trace (CTRACE) options. If this parameter is not specified, the default member CTIIDS00 is used.

Language Environment runtime options and environment variables

These values are used by the stack's Language Environment® functions:

- Configuration
- Autolog
- SNMP TCP/IP Subagent

For example, the TCP/IP stack's configuration function uses the z/OS UNIX search order to locate TCPIP.DATA information to determine the stack's host name. See the search orders used in the z/OS UNIX environment in the z/OS Communications Server: IP Configuration Guide for a description of this search order. Use the RESOLVER_CONFIG environment variable in the PARM= field of the TCP/IP cataloged procedure to specify the TCPIP.DATA file or data set that you want the configuration function to use.

Stack Configuration task tracing parameter, -d or -D

This parameter enables tracing of Configuration task processing before the ITRACE ON CONFIG 1 Profile statement is processed.

Requirement: If this parameter is specified, then it must be the last parameter specified in the PARM= field, and it must be preceded by a slash as in the following example:

```
//TCPIP  EXEC  PARM=('&PARMS',
                // 'ENVAR("RESOLVER_CONFIG=//''TCPIVP.TCPPARMS(TCPDATA)''")'
                // '/ -d')
```

This trace can be disabled by way of a VARY TCPIP,,OBEYFILE command with ITRACE OFF CONFIG statement specified in the data set referenced by the command.

Example of a TCP/IP cataloged procedure

The following code is an example of a TCP/IP cataloged procedure that defines the component tracing and intrusion detection services tracing that is to be in effect for the TCP/IP address space.

```
//TCPIP    PROC PARMS='CTRACE(CTIEZB00),IDS=00'
//*TCPIP    PROC PARMS='TRC=00,IDS=00'
//*
//*  z/OS Communications Server
//*  SMP/E Distribution Name: EZAEB01G
//*
//*  Licensed Materials - Property of IBM
//*  5650-ZOS
//*  Copyright IBM Corp. 1991, 2013
//*  Status = CSV2R1
//*
//* SET PARM1=TCPIVP.TCPPARMS(TCPDATA)
//*
//TCPIP    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,
//             PARM='&PARMS'
//*  Uncomment the SET statement above when using the next two lines.
//* PARM=('&PARMS',
//*       'ENVAR("RESOLVER_CONFIG=//''&PARM1''")')
//*
//* See the TCP/IP cataloged procedure chapter of the IP Configuration
//* Reference for a description of the parameters that can be
//* specified in the PARM= field of the EXEC statement.
//*
//*******************************************************************
//*     The C runtime libraries should be in the system's link list
//*     or add them via a STEPLIB definition here.  If you add
//*     them via a STEPLIB, they must be APF authorized with DISP=SHR
//*
//*STEPLIB  DD ...
//*     Any data set referenced by the STEPLIB DD statement must be
//*     APF authorized.
//*
//*     SYSPRINT contains Resolver run-time diagnostics (TRACE RESOLVER
//*        output).  It can be directed to SYSOUT or a data set.
//*        We recommend directing the output to SYSOUT due to
//*        data set size restraints.
//*     ALGPRINT contains run-time diagnostics from TCP/IP's Autolog
//*        task. It can be directed to SYSOUT or a data set.  We
//*        recommend directing the output to SYSOUT due to data set size
//*        restraints.
//*     CFGPRINT contains run-time diagnostics from TCP/IP's Config
//*        task and TCPIPSTATISTICS counter output.
//*        It can be directed to SYSOUT or a data set. We recommend
//*        directing the output to SYSOUT due to data set size
//*        restraints.
//*     SYSERROR contains console messages issued by TCP/IP's Config
//*        task while processing the initial profile or the data
//*        set specified on a VARY TCPIP,,OBEYFILE command.
//*
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
```

```
//SYSERROR DD SYSOUT=*
//*
//*       TNDBCSCN is the configuration data set for TELNET DBCS
//*       transform mode.
//*
//*TNDBCSCN DD DISP=SHR,
//*           DSN=TCPIP.SEZAINST(TNDBCSCN)
//*
//*       TNDBCSXL contains binary DBCS translation table codefiles
//*       used by TELNET DBCS Transform mode.
//*
//*TNDBCSXL DD DISP=SHR,
//*           DSN=TCPIP.SEZAXLD2
//*
//*       TNDBCSER receives debug output from TELNET DBCS Transform
//*       mode, when TRACE TELNET is specified in the PROFILE data set.
//*
//*TNDBCSER DD SYSOUT=*
//*
//*
//*PROFILE  DD ...
//*     The PROFILE DD statement specifies the data set containing the
//*     TCP/IP configuration parameters.  If the PROFILE DD statement
//*     is not supplied, a default search order is used to find
//*     the PROFILE data set.  See the IP Configuration Guide for
//*     a description of the search order for PROFILE.TCPIP.  A
//*     sample profile is included in member SAMPPROF of the
//*     SEZAINST data set.
//*
//*PROFILE  DD DISP=SHR,
//*           DSN=TCPIVP.TCPPARMS(PROFILE)
//*PROFILE  DD DISP=SHR,
//*           DSN=TCPIP.PROFILE.TCPIP
//*
//*     SYSTCPD explicitly identifies which data set is to be
//*       used to obtain the parameters defined by TCPIP.DATA
//*       when no GLOBALTCPIPDATA statement is configured.
//*       See the IP Configuration Guide for information on
//*       the TCPIP.DATA search order.
//*       The data set can be any sequential data set or a member of
//*       a partitioned data set (PDS).
//*
//*SYSTCPD  DD DISP=SHR,
//*          DSN=TCPIVP.TCPPARMS(TCPDATA)
//*SYSTCPD  DD DISP=SHR,
//*          DSN=TCPIP.SEZAINST(TCPDATA)
```

Figure 3. Sample TCP/IP start up proc

Using output data sets

In the TCP/IP address space, the SYSPRINT and SYSERROR data sets defined with a DD statement must have a variable blocked (VB) format. Block size (BLKSIZE) for a VB RECFM must be at least 4 bytes larger than the logical record length (LRECL).

Guideline: You can allocate these as partitioned or sequential data sets, but be aware that partitioned data sets cannot be reused if they have filled or if the members already exist.

Chapter 4. Protocol number and port assignments

The protocol file or data set is used to map protocol names to protocol numbers. Some applications use getprotobyname() and other socket calls to look up protocol numbers or names. If the protocol file or data set is not present or does not contain the required definitions, certain applications might not function properly.

The sample protocol file or data set provided with z/OS Communications Server and shown in the following example contains the definitions required by most applications. See Chapter 1, "Configuration data sets and files," on page 1 for information about the search order used by the resolvers for locating this file or data set.

Guideline: Keep both *hlq*.ETC.PROTO and /etc/protocol in sync.

```
#
# Licensed Materials - Property of IBM
# 5694-A01
# Copyright IBM Corp. 1995, 2010
# Status = CSV1R12
#
# sample protocol file or dataset, installed in
#
#   /usr/lpp/tcpip/samples/protocol
#   /usr/lpp/tcpip/samples/IBM/EZAOEPRO
#   SEZAINST(PROTO)
#
# Refer to IP Configuration Reference for the search
# order used by the resolver to find this file.
#
# offical name, protocol number, aliases

ip           0               # dummy for IP
hopopt       0               # hop-by-hop options for IPv6
icmp         1               # control message protocol
# ggp        2               # gateway^2 (deprecated)
tcp          6               # tcp
# egp        8               # exterior gateway protocol
# pup        12              # pup
udp          17              # user datagram protocol
# idp        22              # xns idp
ipv6         41              # ipv6
ipv6-icmp    58              # icmpv6
ipv6-route   43  IPv6-Route  # Routing Header for IPv6
ipv6-frag    44  IPv6-Frag   # Fragment Header for IPv6
esp          50              # encapsulating security payload
ipv6-crypt   50  IPv6-Crypt  # Encryption Header for IPv6
ah           51              # Authentication header
ipv6-auth    51  IPv6-Auth   # Authentication Header for IPv6
ipv6-icmp    58  IPv6-ICMP   # ICMP for IPv6
ipv6-nonxt   59  IPv6-NoNxt  # No Next Header for IPv6
ipv6-opts    60  IPv6-Opts   # Destination Options for IPv6
ospf         89              # Open Shortest Path First protocol
```

Figure 4. /etc/proto or ETC.PROTO example

Port assignments

Port numbers are used on various socket calls. They are also included in both the header of a TCP segment and a UDP datagram. You can assign port numbers to your own server applications by adding entries to the z/OS UNIX file or to the data set.

Guidelines:

- Assign ports is by assigning a standard port number and use the Server Bind Control function of the PROFILE.TCPIP PORT statement to assign each server to a separate IP address.
- Use the IP address on the PORT BIND be a VIPA address known to the domain name server (DNS) as a host name that users understand. For example, the RXSERVE procedure is assigned to ports 512 and 514, the orexecd and orshd daemons are assigned to ports 512 and 514, and two IP addresses (host names MVS97 and MVS97USS) 9.67.113.1 and 9.67.113.2 are available.

The following example reflects a situation where more than one application needs to listen on the same port, and the application or applications bind to INADDR_ANY.

In this example, the PORT statement would be as follows:

```
PORT
512 TCP RXSERVE ; Remote Execution Server (default)
512 TCP OMVS BIND 9.67.113.2 ; orexecd Remote Execution Server (MVS97USS)
514 TCP RXSERVE ; Remote Shell Server (default)
514 TCP OMVS BIND 9.67.113.2 ; orshd Remote Shell Server (MVS97USS)
```

Result: Clients who use MVS97 for remote execution get RXSERVE, and clients who use MVS97USS get OMVS orshd.

PROFILE.TCPIP port assignments

Use the PORT and PORTRANGE statement in the PROFILE.TCPIP data set to reserve ports for specified user IDs, procedures, and job names.

Tip: The following example was used for test configuration and is for illustration only. The example shows a portion of SEZAINST(SAMPPROF), which contains the most current assignments.

```
;
PORT: Reserves a port for specified job names
;
;   - A port that is not reserved in this list can be used by any user.
;     If you have TCP/IP hosts in your network that reserve ports
;     in the range 1-1023 for privileged applications, you should
;     reserve them here to prevent users from using them.
;     The RESTRICTLOWPORTS option on TCPCONFIG and UDPCONFIG will also
;     prevent unauthorized applications from accessing unreserved
;     ports in the 1-1023 range.
;
;   - A PORT statement with the optional keyword SAF followed by a
;     1-8 character name can be used to reserve a PORT and control
;     access to the PORT with a security product such as RACF.
;     For port access control, the full resource name for the security
;     product authorization check is constructed as follows:
;     EZB.PORTACCESS.sysname.tcpname.safname
;     where:
;       EZB.PORTACCESS is a constant
;       sysname is the MVS system name (substitute your sysname)
;       tcpname is the TCPIP jobname (substitute your jobname)
;        safname is the 1-8 character name following the SAF keyword
;
;      When PORT access control is used, the TCP/IP application
;      USERID that is authorized to the resource. The resources
;      are defined in the SERVAUTH class.
;
;      For an example of how the SAF keyword can be used to enhance
;      security, see the definition below for the FTP data PORT 20
;      with the SAF keyword. This definition reserves TCP PORT 20 for
;      any jobname (the *) but requires that the FTP user be permitted
;      by the security product to the resource:
;      EZB.PORTACCESS.sysname.tcpname.FTPDATA in the SERVAUTH class.
;
;    - The BIND keyword is used to force a generic server (one that
;      binds to the IPv4 INADDR_ANY address, or the IPv6 unspecified
;      address, in6addr_any) to bind to the specific IP address that
;      is specified following the BIND keyword. This capability could
;      be used, for example, to allow z/OS UNIX telnet and telnet
;      3270 servers to both bind to TCP port 23.
;      The IP address that follows bind must be in IPv4 (dotted
;      decimal) or IPv6 (colon-hexadecimal) format and may be
;      any valid address for the host including VIPA and dynamic
;      VIPA addresses.
;
;    The special jobname of OMVS indicates that the PORT is reserved
;    for any application with the exception of those that use the Pascal
;    API.
;
;    The special jobname of * indicates that the PORT is reserved
;    for any application, including Pascal API socket applications.
;    Jobname may be specified as a prefix of zero to seven characters
;    ending in *.
;
;    The special jobname of RESERVED indicates that the PORT is
;    blocked. It will not be available to any application.
;
;    GUIDELINE: When IPSECURITY is enabled, UDP ports 500 and 4500
;    should either be reserved for IKED (if it is in use) or should
;    be marked RESERVED.
;
;    TIP:  The PORT statement can also be used to control application
;    access to unreserved ports by configuring PORT entries where the
;    port number is replaced by the keyword UNRSV.
;
PORT
     7 UDP MISCSERV            ; Miscellaneous Server - echo
     7 TCP MISCSERV            ; Miscellaneous Server - echo
     9 UDP MISCSERV            ; Miscellaneous Server - discard
     9 TCP MISCSERV            ; Miscellaneous Server - discard
    19 UDP MISCSERV            ; Miscellaneous Server - chargen
    19 TCP MISCSERV            ; Miscellaneous Server - chargen
   20 TCP *  NOAUTOLOG        ; FTP Server
;   20 TCP *  NOAUTOLOG SAF FTPDATA ; FTP Server
    21 TCP FTPD1               ; FTP Server
    23 TCP TN3270              ; Telnet 3270 Server
;   23 TCP INETD1 BIND 9.67.113.3 ; z/OS UNIX Telnet server
    25 TCP SMTP                ; SMTP Server
   111 TCP PORTMAP             ; Portmap Server (SUN 3.9)
   111 UDP PORTMAP             ; Portmap Server (SUN 3.9)
;  111 TCP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
;  111 UDP PORTMAP1            ; Unix Portmap Server (SUN 4.0)
   123 UDP SNTPD               ; Simple Network Time Protocol Server
   135 UDP LLBD                ; NCS Location Broker
   161 UDP OSNMPD              ; SNMP Agent
   389 TCP LDAPSRV             ; LDAP Server
   443 TCP HTTPS               ; http protocol over TLS/SSL
   443 UDP HTTPS               ; http protocol over TLS/SSL
```

```
;  500 UDP IKED                ; CS IKE daemon
   512 TCP RXSERVE             ; Remote Execution Server
   514 TCP RXSERVE             ; Remote Execution Server
;  512 TCP * SAF OREXECD       ; z/OS UNIX Remote Execution Server
;  514 TCP * SAF ORSHELLD      ; z/OS UNIX Remote Shell Server
;  515 TCP LPSERVE             ; LPD Server
;  515 TCP AOPLPD              ; Infoprint LPD Server
   520 UDP OMPROUTE            ; OMPROUTE Server (IPv4 RIP)
   521 UDP OMPROUTE            ; OMPROUTE Server (IPv6 RIP)
   580 UDP NCPROUT             ; NCPROUTE Server
   750 TCP MVSKERB             ; Kerberos
   750 UDP MVSKERB             ; Kerberos
   751 TCP ADM@SRV             ; Kerberos Admin Server
   751 UDP ADM@SRV             ; Kerberos Admin Server
; 1700 TCP PAGENT NOAUTOLOG    ; Policy Agent pagentQosListener port
; 1701 TCP PAGENT NOAUTOLOG    ; Policy Agent pagentQosCollector port
  3000 TCP CICSTCP             ; CICS Socket
  3389 TCP MSYSLDAP            ; LDAP Server for Msys
; 4159 TCP NSSD                ; CS NSS daemon
; 4500 UDP IKED                ; CS IKE daemon
;16310 TCP PAGENT NOAUTOLOG    ; Policy Agent server listener port
;
;
; PORTRANGE: Reserves a range of ports for specified jobnames.
;
;   In a common INET (CINET) environment, the port range indicated by
;   the INADDRANYPORT and INADDRANYCOUNT in your BPXPRMxx parmlib member
;   should be reserved for OMVS.
;
;   The special jobname of OMVS indicates that the PORTRANGE is reserved
;   for ANY z/OS UNIX socket application.
;
;   The special jobname of * indicates that the PORTRANGE is reserved
;   for any socket application, including Pascal API socket
;   applications.
;
;   The special jobname of RESERVED indicates that the PORTRANGE is
;   blocked. It will not be available to any application.
;
;   The SAF keyword is used to restrict access to the PORTRANGE to
;   authorized users. See the use of SAF on the PORT statement above.
;
;
;   PORTRANGE 4000 1000 TCP OMVS
;   PORTRANGE 4000 1000 UDP OMVS
;   PORTRANGE 2000 3000 TCP RESERVED
;   PORTRANGE 5000 6000 TCP * SAF RANGE1
;
```

Figure 5. Sample TCP/IP start up proc

/etc/services and ETC.SERVICES port assignments

The z/OS UNIX file, /etc/services, contains the service names and port assignments of specific z/OS UNIX applications. The MVS data set ETC.SERVICES can also be used to contain the same information. The source for this example is shipped in SEZAINST(SERVICES) and copied to the *hlq*.ETC.SERVICES by the Installation Verification Procedure (IVP). The source is also installed in /usr/lpp/tcpip/samples/services for use in copying it to /etc/services. It is important that /etc/services and *hlq*.ETC.SERVICES be kept identical so that MVS and z/OS UNIX applications use the same port assignments. The shipped file contains the most current assignments.

Rules: The following syntax rules apply to the services information specification:

- An ETC.SERVICES data set must be fixed or fixed block with an LRECL between 56 and 256.
- The /etc/services z/OS UNIX file can have a maximum line length of 256.
- Each service is listed on a single line corresponding to the form:

  ```
  ServiceName   PortNumber/ProtocolName   Aliases
  ```

  ServiceName
  Specifies an official Internet service name.

  PortNumber
  Specifies the socket port number used for the service.

  ProtocolName
  Specifies the transport protocol used for the service.

  Aliases
  Specifies a list of unofficial service names.

  Items on a line are separated by spaces or tabs.
- A service name must start in the first position on a line.
- The maximum service name and alias name length is 32 characters.
- A maximum of 35 aliases is recognized.
- Service and alias names are case sensitive.
- Comments begin with a # or ; character and continue until the end of the line.

When services information is requested, the definitions are searched sequentially. The first entry matching a specified search request (either service name and protocol or port number and protocol) is returned.

For the search order used in locating /etc/services and ETC.SERVICES, see z/OS Communications Server: IP Configuration Guide.

Tip: The following example was used for test configuration and is for illustration only.

```
#  z/OS Communications Server
#  SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZAOESER
#  SMP/E distribution path: SEZAINST(EZAEB02J)
#
#  5694-A01 Copyright IBM Corp. 1998, 2009
#  Licensed Materials - Property of IBM
#  Status = CSV1R11
#
# $Header:services 9.4$
# $ACIS:services 9.4$
# $Source: /ibm/acis/usr/src/etc/RCS/services,v $
#  Change Activity:
#
#  Flag Reason   Release  Date   Origin    Description
#  ---- -------- -------- ------ --------  -----------------------
#  $X1= D136984  R8BASEN  060303 ADAMSON : JES NJE over TCP/IP
#  $Y1= D139394  R9BASEN  061011 AMITRANO: NFS port information
#  $F1= D146073  RBBASE   081016 PACKETTA: Add port 521 alias
# ----------------------------------------------------------------

#      @(#)services    1.16 (Berkeley) 86/04/20
#
# Network services, Internet style
#
# Service     port/protocol   alias names if any                 @Y1A
#
echo           7/tcp
echo           7/udp
discard        9/tcp           sink null
discard        9/udp           sink null
```

```
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
tftp            69/udp
rje             77/tcp          netrjs
finger          79/tcp
link            87/tcp          ttylink
supdup          95/tcp
hostnames       101/tcp         hostname        # usually from sri-nic
#csnet-cs       105/?
pop             109/tcp         postoffice
sunrpc          111/tcp
sunrpc          111/udp
auth            113/tcp         authentication
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Xfer Proto
ntp             123/udp                         # Network Time Protocol
snmp            161/udp                         # snmp request port
snmp-trap       162/udp                         # snmp monitor trap port
vmnet           175/tcp         # JES NJE over TCP/IP            @X1A
#
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
ntalk           518/udp
efs             520/tcp                         # for LucasFilm
#
# IBM added services
#
route           520/udp         router omproute
route           521/udp         ipv6rip ripng   #                @F1C

timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc
conference      531/tcp         chat
#
# RVD service
#
rvd-control     531/udp                         # rvd control port
netnews         532/tcp         readnews
netwall         533/udp                      # -for emergency broadcasts
uucp            540/tcp         uucpd           # uucp daemon
#
#     Kerberos services
#
klogin          543/tcp           # Kerberos authenticated rlogin
kshell          544/tcp      cmd  # Kerberos remote shell

remotefs        556/tcp     rfs_server rfs  # Brunhoff remote filesystem
#
# IBM added service
#
ncprout         580/udp         ncproute
#
# Andrew File System Authenticated services
#
```

```
vexec           712/tcp         vice-exec
vlogin          713/tcp         vice-login
vshell          714/tcp         vice-shell
#
#     Kerberos services
#
kerberos        750/udp      kdc  # Kerberos authentication--udp
kerberos        750/tcp      kdc  # Kerberos authentication--tcp
kerboros_master 751/udp           # Kerberos authentication
kerberos_master 751/tcp           # Kerberos authentication
passwd_server   752/udp           # Kerberos passwd server
userreg_server  753/tcp           # Kerberos userreg server
krb_prop        754/tcp           # Kerberos slave propagation
erlogin         888/tcp           # Login and environment passing
#
#
#       Kerberos sample server
#
sample          906/tcp           # Kerberos sample app server
sample          906/udp           # for kerberos simple test

kpop            1109/tcp          # Pop with Kerberos

ingreslock      1524/tcp
#
#       Policy Agent QoS Listener and Collector ports
#
pagentQosListener  1700/tcp       # Policy Agent Listener thread
pagentQosCollector 1701/tcp       # Policy Agent Collector thread
#
# Andrew File System services
#
filesrv         2001/tcp
rauth2          2001/udp
rfilebulk       2002/udp
rfilesrv        2003/udp
console         2018/udp
# For file server backup and migration
client          2030/tcp
# NFS server                                                  @Y1A
#                                                             @Y1A
#  Port 2049 must be used for nfsd.                           @Y1A
#                                                             @Y1A
#  Consecutive port numbers must be assigned for the NFS status,    @Y1A
#   nlockmgr, mountd, mvsmount, showattr, and pcnfsd services.      @Y1A
#   The example below uses ports 2043-2048.                   @Y1A
#  When the NFS callback function is being used the services  @Y1A
#   nfsscb_b and nfsscb_e should reserve 100 consecutive ports.     @Y1A
#   The example below uses port 10300 for the beginning port  @Y1A
#   and port 10399 as the ending port.                        @Y1A
#  For additional information see the Network File System Guide     @Y1A
#   and Reference manual.                                     @Y1A
#                                                             @Y1A
status          2043/tcp   nfs_statd     # NFS State daemon (NSM)   @Y1A
status          2043/udp   nfs_statd     # NFS State daemon (NSM)   @Y1A
nlockmgr        2044/tcp   nfs_lockd     # NFS Lock daemon (NLM)    @Y1A
nlockmgr        2044/udp   nfs_lockd     # NFS Lock daemon (NLM)    @Y1A
mountd          2045/tcp   mount         # NFS mount daemon         @Y1A
mountd          2045/udp   mount         # NFS mount daemon         @Y1A
mvsmount        2046/tcp   nfs_mvsmnt    # NFS mvsmount daemon      @Y1A
mvsmount        2046/udp   nfs_mvsmnt    # NFS mvsmount daemon      @Y1A
showattr        2047/tcp   nfs_showattr  # NFS showattr daemon      @Y1A
showattr        2047/udp   nfs_showattr  # NFS showattr daemon      @Y1A
pcnfsd          2048/udp   nfs_pcnfs     # NFS pcnfsd daemon        @Y1A
nfsd            2049/tcp   nfs           # NFS server daemon        @Y1A
                                         #  - do not change         @Y1A
nfsd            2049/udp   nfs           # NFS server daemon        @Y1A
                                         #  - do not change         @Y1A
#                                                             @Y1A
#  NFS Callback function port range                           @Y1A
#                                                             @Y1A
nfsscb_b       10300/tcp                 # NFSS callback port begin @Y1A
nfsscb_e       10399/tcp                 # NFSS callback port end   @Y1A
nfsscb_b       10300/udp                 # NFSS callback port begin @Y1A
nfsscb_e       10399/udp                 # NFSS callback port end   @Y1A
#
#     Kerberos services
#
knetd           2053/tcp          # Kerberos de-multiplexor
eklogin         2105/tcp          # Kerberos encrypted rlogin
```

```
#
# Andrew File System services
#
venus.itc       2106/tcp
ropcons         2115/udp
# The following are assigned in pairs and the bulk must be the srv +1
rupdsrv         2131/udp
rupdbulk        2132/udp
rupdsrv1        2133/udp
rupdbulk1       2134/udp

njenet-ssl      2252/tcp          # JES NJE over TCP/IP with SSL @X1A
```

Figure 6. /etc/services example

Chapter 5. Resolver setup and TCPIP.DATA configuration statements

This topic contains the following information:

- "Resolver setup statements"
- "Sample TCPIP.DATA data set (TCPDATA)" on page 398

Resolver setup statements

The resolver address space can be customized with the following resolver setup statements summarized in Table 7.

See z/OS Communications Server: IP Configuration Guide for more information about resolvers.

Table 7. Summary of resolver setup statements

Statement	Description	See
CACHE/ NOCACHE	CACHE indicates that system-wide caching is enabled for the resolver. NOCACHE indicates that system-wide caching is not enabled for the resolver.	"CACHE NOCACHE statements" on page 358
CACHESIZE	CACHESIZE specifies the maximum amount of storage that can be allocated by the resolver to manage cached records. **Tip:** CACHESIZE is ignored unless CACHE is also specified.	"CACHESIZE statement" on page 359
COMMONSEARCH/ NOCOMMONSEARCH	COMMONSEARCH indicates that the search order for local host tables is the same regardless of whether the query is for IPv6 or IPv4 addresses. The search order is also the same regardless of whether the query is issued under the native MVS or the z/OS UNIXenvironment. NOCOMMONSEARCH indicates that the search order for local host tables is different for IPv4 and IPv6 queries. The search order is also different for queries issued under the native MVS environment, vs. queries issued under the z/OS UNIX environment.	"COMMONSEARCH/NOCOMMONSEARCH statement" on page 360

Table 7. Summary of resolver setup statements (continued)

Statement	Description	See
DEFAULTIPNODES	Specifies the name of either a z/OS UNIX file or MVS data set that contains the hard-coded IP addresses and host names to be used. Identifies the default search location for IPNODES local host file.	"DEFAULTIPNODES statement" on page 360
DEFAULTTCPIPDATA	Specifies the name of either a z/OS UNIX file or MVS data set that contains the TCPIP.DATA statement that is used instead of TCPIP.TCPIP.DATA as the final location when searching for TCPIP.DATA.	"DEFAULTTCPIPDATA statement" on page 361
GLOBALIPNODES	Specifies the name of either a z/OS UNIX file or MVS data set that contains the hard-coded IP addresses and host names to be used. Identifies the first search location for IPNODES local host file.	"GLOBALIPNODES statement" on page 362
GLOBALTCPIPDATA	Specifies the name of either a z/OS UNIX file or MVS data set that contains the TCPIP.DATA statement that is used to set global MVS image-wide values for TCPIP.DATA.	"GLOBALTCPIPDATA statement" on page 363
MAXTTL	Specifies the maximum amount of time the resolver can use resource information obtained from a Domain Name System (DNS) server as part of resource resolution.	"MAXTTL statement" on page 365
UNRESPONSIVETHRESHOLD	Specify the threshold value for when a name server is declared to be unresponsive to resolver queries. Also specifies whether the resolver automatically stops sending DNS queries generated by an application to an unresponsive name server.	"UNRESPONSIVETHRESHOLD statement" on page 366
; or #	Indicates a comment.	"; and # statements" on page 368

Resolver setup statement information and syntax conventions

This topic explains each of the resolver setup statements in detail.

If resolver setup statements are contained in a data set, the data set can have the following characteristics:

- Sequential (PS) or partitioned (PO) organization

- Fixed (F) or fixed block (FB) format
- Recommended logical record length (LRECL) in the range 80 - 256
- Any valid block size

Restriction: If resolver setup statements are contained in a z/OS UNIX file, the file can have a maximum line length of 256.

Observe the following syntax conventions for resolver setup statements:

- A blank indicates the end of a statement's values. Anything following the blank on the same line is treated as a comment.
- Static system symbols can be used in resolver setup file statements.
- If a valid statement has any parameter error, a warning message is displayed on the operator console and in the JES job log.
  - If the statement was found during resolver address space initialization, the statement is ignored. Processing of the setup statements continues.
  - If the statement was found while processing a MODIFY RESOLVER,REFRESH,SETUP command, processing of the setup statements ends, the MODIFY fails, and no refresh takes place.
- If an invalid statement is found, a warning message is displayed on the operator's console and in the JES job log.
  - If the invalid statement was found during resolver address space initialization, the statement is ignored. Processing of the setup statements continues.
  - If the invalid statement was found while processing a MODIFY RESOLVER,REFRESH,SETUP command, processing of the setup statements ends, the MODIFY fails, and no refresh takes place.
- When setup file processing is complete, the resolver attempts to open the MVS data set or z/OS UNIX file name specified as the parameter on the last valid instance of the GLOBALIPNODES, DEFAULTIPNODES, GLOBALTCPIPDATA, and DEFAULTTCPIPDATA setup statements. If the parameter value was incorrect or the specified z/OS UNIX or MVS data set does not exist, a warning message is displayed on the operator console and in the JES job log.
  - If the error occurred during resolver address space initialization, processing continues. The resolver assumes the default setting for the setup statement, which in this case is no file was specified.
  - If the error occurred while processing a MODIFY RESOLVER,REFRESH,SETUP command, processing of the setup statements ends, the MODIFY fails, and no refresh takes place.
- When the setup file processing successfully completes, each resolver statement's value is displayed on the operator console and in the JES job log. If a resolver statement can specify an MVS data set or z/OS UNIX file and none was specified, the word NONE is displayed as the statement's value on the operator console and in the JES job log.
- The resolver processes an MVS data set or z/OS UNIX file specified on the GLOBALTCPIPDATA resolver setup statement differently than data sets or files that are specified on other resolver setup statements:
  - During the initialization of the resolver address space and while processing the MODIFY,REFRESH command, the resolver reads the contents of the MVS data set or the z/OS UNIX file that is specified on the GLOBALTCPIPDATA resolver setup statement. The resolver saves the information for resolver NMI purposes. See Resolver NMI (EZBREIFR) in z/OS Communications Server: IP Programmer's Guide and Reference for details.

  - For all other data set or file specifications, the contents of an MVS data set or z/OS UNIX file that is specified by a resolver setup statement are not validated during the initialization of the resolver address space or while the MODIFY,REFRESH command is processing. The contents are validated when the first usage of resolver services is requested by an address space.
- If the resolver address space abnormally terminates, an eventual action message is issued which indicates the failure. Use the START operator command to restart the Resolver address space.
- Resolver initialization deletes any resolver-related eventual action messages.
- If an allocation error occurs when trying to access a resolver statement's MVS data set or z/OS UNIX file, an eventual action message is issued to the operator console. Only one eventual action message per MVS data set or z/OS UNIX file is issued regardless of the number of times the file is accessed. After a successful reference to the file has occurred, the message is removed from the operator console.
  - If the allocation error occurs during resolver address space initialization, processing continues.
  - If the allocation error occurs while resolver is processing a MODIFY RESOLVER,REFRESH command, processing of the setup statements ends and the MODIFY command fails. No refresh takes place.
  - If the allocation error occurs during a resolver API call, processing of the resolver API continues. The resolver takes default values, if appropriate, for any statements that could have appeared in the data set or file.
- If no errors are detected during resolver address space initialization, the resolver issues message EZZ9291I at the completion of initialization processing. If an error is detected, the resolver issues message EZD2038I. You can create automation to look for message EZD2038I and to alert you about errors in your resolver setup file that might cause the resolver to behave differently than intended.

To determine the current setting of the resolver setup statements, use the MODIFY RESOLVER,DISPLAY operator console command. See z/OS Communications Server: IP System Administrator's Commands for MODIFY command.

For more information about resolvers, see z/OS Communications Server: IP Configuration Guide

CACHE NOCACHE statements

Use the CACHE statement to enable system-wide caching for the resolver. Use the NOCACHE statement to disable system-wide caching.

System-wide caching saves resource information obtained from name servers during processing of application queries, which permits subsequent queries for the same resource to be satisfied without contacting the name server for the information. The resolver caches both positive and negative resource information. For more information about the resolver caching function, see z/OS Communications Server: IP Configuration Guide.

The default value is CACHE.

Syntax

```
   ┌─CACHE───┐
►►─┴─NOCACHE─┴──────────────────────────────────────────────────────────►◄
```

Parameters

This statement has no parameters.

CACHESIZE statement

Use the CACHESIZE statement to specify the maximum amount of storage that can be allocated by the resolver to manage cached records.

Tip: When the CACHESIZE statement is specified with the NOCACHE statement, the CACHESIZE operand is ignored.

Syntax

```
   ┌─CACHESIZE(200M)──────────┐
►►─┴─CACHESIZE(cachesizeM)────┴─────────────────────────────────────────►◄
```

Parameters

`cachesize M`
: Specifies the maximum amount of 64-bit private virtual storage that the resolver can use to maintain cache information. This limit should be expressed as a number followed by an M (which represents 1 048 576 bytes). The *cachesize* value must be in the range 1M - 999M. The default is 200M.

Examples

Use the CACHESIZE statement to set 10 M as the maximum cache size value:

```
CACHESIZE(10M)
```

Usage notes

- For planning purposes, 1 megabyte of storage can contain between 400 and 450 cache entries. The actual values can vary depending on the amount of storage needed to hold cache infrastructure control blocks used to represent the entries and the name servers providing the information.
- The resolver acquires cache storage incrementally as needed, up to the maximum specified by the CACHESIZE operand. Because storage is acquired incrementally, there is no penalty for specifying a CACHESIZE value significantly greater than the expected maximum amount of storage required. This avoids storage constraint processing during spikes in the amount of information being cached. Consider specifying a value at least fifty percent larger than the amount of storage you actually expect to be used for cache entries.
- You can increment the value of *cachesize* M dynamically by issuing the MODIFY RESOLVER,REFRESH,SETUP=*resolver_setup_filename* command. You cannot decrease the value of *cachesize* M dynamically. If you attempt to lower the value

of CACHESIZE dynamically, the MODIFY command fails and message EZZ9306I is issued. To decrease the value of *cachesize* M, you must stop and restart the resolver. Alternatively, update the resolver setup file to indicate NOCACHE and issue a MODIFY RESOLVER,REFRESH,SETUP=*resolver_setup_filename* command to first stop resolver caching, and then issue a second MODIFY RESOLVER,REFRESH,SETUP=*resolver_setup_filename* command that both restarts resolver caching and decrease the value of *cachesize* M.

COMMONSEARCH/NOCOMMONSEARCH statement

Use the COMMONSEARCH statement to indicate that the search order for local host table is the same regardless of whether it is for an IPv6 or an IPv4 query, or whether the query is issued in the native MVS or z/OS UNIX environment. The default is NOCOMMONSEARCH.

Restriction: You must code the COMMONSEARCH statement if you use IPNODES for the local hosts file. For more information, see Search orders used in the z/OS UNIX environment and Search orders used in the native MVS environment in z/OS Communications Server: IP Configuration Guide.

Syntax

```
      ┌─NOCOMMONSEARCH─┐
►►────┼────────────────┼─────────────────────────────────────────────────►◄
      └─COMMONSEARCH───┘
```

Parameters

This statement has no parameters.

Examples

To code COMMONSEARCH:

```
COMMONSEARCH
```

DEFAULTIPNODES statement

Use the DEFAULTIPNODES statement to specify the name of either a z/OS UNIX file or MVS data set that contains the hard-coded IP addresses and host names to be used.

Restriction: The specified file or data set can contain IPv4 and IPv6 addresses, but cannot contain IPv4–mapped addresses.

Syntax

```
►►────┬──────────────────────────────────────────────────────────┬───────►◄
      └─DEFAULTIPNODES ──┬─('fully qualified MVS dataset name')─┘
                         └─(/file system absolute pathname)───┘
```

Parameters

`'fully qualified MVS dataset name'`
The complete name of the MVS data set containing the IP addresses and host names. The data set name is not case sensitive.

Requirement: The beginning and ending quotation marks (' ') are required.

`/file system absolute pathname`
The complete name of the z/OS UNIX file containing the IP addresses and host names. The z/OS UNIX path name is case sensitive.

Requirement: The beginning slash (/) is required.

Restriction: The /file system absolute path name can be a maximum of 255 characters.

Examples

To specify the data set named TCPIP.ETC.IPNODES, use the following code:

```
DEFAULTIPNODES('TCPIP.ETC.IPNODES')
```

To specify z/OS UNIX file ipnodes in directory etc as containing IP addresses and host names, use the following code:

```
DEFAULTIPNODES(/etc/ipnodes)
```

Note: Because it is a z/OS UNIX file, the name is case sensitive.

Usage notes

- For a z/OS UNIX file, the file can reside in any directory. The maximum line length supported is 256 characters. If the line is greater than 256 characters, it is truncated to 256 characters and processed, and a trace resolver warning message is issued.
- For an MVS data set, the following conditions are required:
  - Sequential (PS) organization or PDS
  - RECFM=F or RECFM=FB
  - A logical record length (LRECL) in the range 80 - 256
- This specified file or data set can include IPv4 and IPv6 addresses, but cannot include IPv4–mapped addresses. Each host name can be up to 128 characters in length, and each host name can have up to 35 IPv4 addresses and 35 IPv6 addresses. Each node in the host name (without dots) can be up to 63 characters in length. For example, if host name is testname.testdomain, *testname* and *testdomain* can be up to 63 characters in length.

DEFAULTTCPIPDATA statement

Use the DEFAULTTCPIPDATA statement to specify the name of either a z/OS UNIX file or MVS data set that contains the TCPIP.DATA statements. This name is used, instead of TCPIP.TCPIP.DATA, as the final location when searching for TCPIP.DATA statements.

Syntax

```
►►──┬─────────────────────────────────────────────────────────────┬────────────►◄
    └─DEFAULTTCPIPDATA ──┬─('fully qualified MVS dataset name')─┬─┘
                         └─(/file system absolute pathname)─────┘
```

Parameters

'fully qualified MVS dataset name'
: The complete name of the MVS data set containing the TCPIP.DATA statements.

  Requirement: The beginning and ending quotation marks (' ') are required.

/file system absolute pathname
: The complete name of the z/OS UNIX file containing the TCPIP.DATA statements.

  Requirement: The beginning slash (/) is required.

  Restriction: The /file system absolute path name can be a maximum of 255 characters.

Examples

The following example specifies member RESLVCF in partitioned data set TCPIP.TCPPARMS as containing TCPIP.DATA statements.

```
DEFAULTTCPIPDATA('TCPIP.TCPPARMS(RESLVCF)')
```

The following example specifies z/OS UNIX file DefaultTcpip.data in directory etc as containing TCPIP.DATA statements.

Note: Because it is a z/OS UNIX file, the name is case sensitive.

```
DEFAULTTCPIPDATA(/etc/DefaultTcpip.data)
```

Usage notes

- For a z/OS UNIX file, the file can reside in any directory. The maximum line length supported is 256 characters. If the line is greater than 256 characters, it is truncated to 256 characters and processed, and a trace resolver warning message is issued.
- The z/OS UNIX path name is case sensitive.
- For an MVS data set, the following conditions are required:
  - Sequential (PS) or Partitioned (PO) organization
  - RECFM=F or RECFM=FB
  - Recommended logic record length (LRECL) in the range 80 - 256
- The MVS data set name is not case sensitive.

GLOBALIPNODES statement

Use the GLOBALIPNODES statement to specify the name of either a z/OS UNIX file or MVS data set that contains the hard-coded IP addresses and host names to be used.

Restriction: The specified file or data set can include IPv4 and IPv6 addresses, but cannot include IPv4–mapped addresses.

Syntax

```
►►──┬─────────────────────────────────────────────────────────┬──────────────►◄
    └─GLOBALIPNODES ──┬─('fully qualified MVS dataset name')─┤
                      └─(/file system absolute pathname)─────┘
```

Parameters

'fully qualified MVS dataset name'
The complete name of the MVS data set containing the IP addresses and host names. The data set name is not case sensitive.

Requirement: The beginning and ending quotation marks (' ') are required.

/file system absolute pathname
The complete name of the file system containing the IP addresses and host names. The z/OS UNIX path name is case sensitive.

Requirement: The beginning slash (/) is required.

Restriction: The /file system absolute path name can be a maximum of 255 characters.

Examples

To specify the data set named TCPIP.ETC.IPNODES, use the following code:

```
GLOBALIPNODES('TCPIP.ETC.IPNODES')
```

To specify z/OS UNIX file **ipnodes** in directory etc as containing IP addresses and host names, use the following code:

```
GLOBALIPNODES(/etc/ipnodes)
```

Note: Because it is an z/OS UNIX file, the name is case sensitive.

Usage notes

- For a z/OS UNIX file, the file can reside in any directory. The maximum line length supported is 256 characters. If the line is greater than 256 characters, it is truncated to 256 characters and processed, and a trace resolver warning message is issued.
- For an MVS data set, the following conditions are required:
  - Sequential (PS) organization or PDS
  - RECFM=F or RECFM=FB
  - Recommended logic record length (LRECL) in the range 80 - 256
- This specified file or data set can contain IPv4 and IPv6 addresses, but cannot contain IPv4–mapped addresses. Each host name can be up to 128 characters in length, and each host name can have up to 35 IPv4 addresses and 35 IPv6 addresses. Each node in the host name (without dots) can be up to 63 characters in length. For example, if host name is testname.testdomain, *testname* and *testdomain* can be to 63 characters in length.

GLOBALTCPIPDATA statement

Use the GLOBALTCPIPDATA statement to specify the name of either a z/OS UNIX file or MVS data set that contains the TCPIP.DATA statements that are used to set global MVS image-wide values for TCPIP.DATA.

If GLOBALTCPIPDATA is not specified, the appropriate environment's (Native MVS or z/OS UNIX) search order is used to locate TCPIP.DATA.

If GLOBALTCPIPDATA is specified, any TCPIP.DATA statements contained in the specified file or data set take precedence over any TCPIP.DATA statements found using the appropriate environment's (native MVS or z/OS UNIX) search order.

The following resolver TCPIP.DATA statements can be specified only in the file or data set specified by GLOBALTCPIPDATA. If these resolver statements are found in any of the other search locations for TCPIP.DATA, they are ignored. If these resolver statements are not found in the file or data set specified by GLOBALTCPIPDATA, their default value is used.

- DomainOrigin/Domain
- NSInterAddr/NameServer
- NSPortAddr
- ResolverTimeOut
- ResolverUDPRetries
- ResolveVia
- Search
- SortList

Syntax

```
►►─┬───────────────────────────────────────────────────────────────┬─────────────►◄
   └─GLOBALTCPIPDATA ──┬─('fully qualified MVS dataset name')─┬─┘
                       └─(/file system absolute pathname)─────┘
```

Parameters

'fully qualified MVS dataset name'
The complete name of the MVS data set containing the TCPIP.DATA statements.

Requirement: The beginning and ending quotes (' ') are required.

/file system absolute pathname
The complete name of the z/OS UNIX file containing the TCPIP.DATA statements.

Requirement: The beginning slash (/) is required.

Restriction: The /file system absolute path name can be a maximum of 255 characters.

Examples

The following example specifies member GLOBAL in partitioned data set TCPIP.TCPPARMS as containing TCPIP.DATA statements.

```
GLOBALTCPIPDATA('TCPIP.TCPPARMS(GLOBAL)')
```

The following example specifies z/OS UNIX file Global.Tcpip.data in directory etc as containing TCPIP.DATA statements.

Note: Because it is a z/OS UNIX file the name is case sensitive.

```
GLOBALTCPIPDATA(/etc/Global.Tcpip.data)
```

Usage notes

- For a z/OS UNIX file, the file can reside in any directory. The maximum line length supported is 256 characters. If the line is greater than 256 characters, it is truncated to 256 characters and processed, and a trace resolver warning message is issued.
- The z/OS UNIX path name is case sensitive.
- For an MVS data set, the following conditions are required:
  - Sequential (PS) or Partitioned (PO) organization
  - RECFM=F or RECFM=FB
  - Recommended logic record length (LRECL) in the range 80 - 256
- The MVS data set name is not case sensitive.
- You must code the GLOBALTCPIPDATA statement if you specify the AUTOQUIESCE operand on the UNRESPONSIVETHRESHOLD statement. If you cannot ensure that all DNS IP addresses are accessible from all of your TCPIP stacks, you should not use a global TCPIP.DATA file and you should not code AUTOQUIESCE on the UNRESPONSIVETHRESHOLD setup statement in your resolver setup file. See "UNRESPONSIVETHRESHOLD statement" on page 366 for more information.

MAXTTL statement

Use the MAXTTL statement to specify the maximum amount of time the resolver can use resource information obtained from a Domain Name System (DNS) server as part of resource resolution.

Tip: When MAXTTL is specified with NOCACHE, the value is ignored.

Syntax

```
     ┌─MAXTTL(2147483647)───┐
►►───┼──────────────────────┼──────────────────────────────────────────────►◄
     └─MAXTTL(time-to-live)─┘
```

Parameters

time-to-live
: Specifies the maximum amount of time, in seconds, that the resolver is permitted to use cached information about a resource obtained from a name server. The *time-to-live* value must be in the range 1 - 2 147 483 647. The default is 2 147 483 647, which is the largest value that can be specified for the *time-to-live* value for a resource at a name server. Specifying, or defaulting to, a value of 2 147 483 647 means that the resolver uses the *time-to-live* value received from the name server to determine how long the resource information can be used.

Examples

The following code is an example of coding 10 minutes (or 600 seconds) as the maximum *time-to-live* value by using the MAXTTL statement:

```
MAXTTL(600)
```

Usage notes

- You can change the value for MAXTTL dynamically using the MODIFY RESOLVER,REFRESH,SETUP=*resolver_setup_filename* command. Changing the value of MAXTTL dynamically has no affect on existing records in the resolver cache, but the new value is used for cache records created after the MODIFY command completes.
- The value for MAXTTL applies to both negative cache records and cache records that represent successful resolution attempts.
- If the time-to-live value that is received from the name server for a given resource is lower than the MAXTTL value, the cached entry times out based on the name server TTL value.
- If the time-to-live value that is received from the name server for a given resource is higher than the MAXTTL value, the cached entry times out based on the MAXTTL value

UNRESPONSIVETHRESHOLD statement

Use the UNRESPONSIVETHRESHOLD statement to specify the threshold value for when the resolver name server monitoring function declares a DNS name server to be unresponsive to resolver queries, and to specify whether the resolver should automatically stop using unresponsive name servers for Domain Name System (DNS) queries generated by an application. For detailed information about the name server monitoring function and about selecting an appropriate value for UNRESPONSIVETHRESHOLD, see the information about monitoring the responsiveness of Domain Name System name servers in z/OS Communications Server: IP Configuration Guide.

Syntax

```
    ┌─UNRESPONSIVETHRESHOLD(25)─────────────────────────────────────┐
►►──┼───────────────────────────────────────────────────────────────┼──────────────►◄
    ├─UNRESPONSIVETHRESHOLD(percentage_of_queries)──────────────────┤
    └─UNRESPONSIVETHRESHOLD(percentage_of_queries,AUTOQUIESCE)──────┘
```

Parameters

percentage_of_queries
: The threshold value for determining when the resolver declares a DNS name server to be unresponsive. The threshold represents a percentage of failures within a specific time interval. The duration of the time interval depends on the setting of the AUTOQUIESCE operand:

  - If you do not specify the AUTOQUIESCE operand, the resolver monitors name server responsiveness using sliding 5-minute intervals.
  - If you specify the AUTOQUIESCE operand, the resolver monitors name server responsiveness using 30-second intervals.

  Valid values for *percentage_of_queries* are in the range 0-100.

  The value 0 indicates that the resolver should not monitor name server responsiveness.

  The default percentage depends on the setting of the AUTOQUIESCE operand:

  - If you do not specify the AUTOQUIESCE operand, the default percentage value is 25% of resolver queries.

- If you specify the AUTOQUIESCE operand, there is no default percentage. You must specify a percentage value when specifying the AUTOQUIESCE operand.

AUTOQUIESCE

Indicates what actions the resolver takes when it identifies an unresponsive name server.

- If you do not specify the AUTOQUIESCE operand and a name server fails to respond to a percentage of resolver queries that is equal to or greater than the specified threshold level, the resolver displays message EZZ9308E to the operator console to indicate that the name server has been unresponsive for this latest 5-minute interval. The resolver continues to forward DNS queries generated by an application to the name server whether it is responsive or not. The name server is considered to be unresponsive until the percentage of queries that the name server does not respond to is less than the specified threshold, or until monitoring is disabled.
- If you specify the AUTOQUIESCE operand and a name server fails to respond to a percentage of resolver queries that is equal to or greater than the specified threshold level, the resolver displays message EZZ9311E to the operator console to indicate that the name server has been unresponsive for this latest 30-second interval. The resolver stops sending DNS queries generated by an application to the name server while the name server is unresponsive. While a name server is unresponsive, the resolver periodically sends DNS polling queries to the name server. The name server is considered to be unresponsive until the percentage of DNS polling queries that the name server does not respond is less than the specified threshold, or until the monitoring is disabled. In most cases, a minimum of 10 DNS polling queries must be attempted before the resolver considers a name server to be responsive.

  Result: If all name servers specified in the global TCPIP.DATA file are unresponsive, the resolver sends DNS queries generated by an application to those name servers, rather than failing the DNS query immediately.

The AUTOQUIESCE operand is used only when the *percentage_of_queries* value is greater than 0.

If you specify the AUTOQUIESCE operand, you must also code the GLOBALTCPIPDATA statement. If you do not code the GLOBALTCPIPDATA statement, the resolver issues message EZD2036I and ignores the AUTOQUIESCE operand. If you cannot ensure that all DNS IP addresses are accessible from all of your TCPIP stacks, you should not use a global TCPIP.DATA file and you should not code AUTOQUIESCE on the UNRESPONSIVETHRESHOLD setup statement in your resolver setup file. See “GLOBALTCPIPDATA statement” on page 363 for details on coding the GLOBALTCPIPDATA statement.

The default setting is that AUTOQUIESCE is not specified.

Steps for modifying

You can refresh this statement using the MODIFY command. To modify the threshold value, the setting of the AUTOQUIESCE operand, or both, perform the following steps:

1. If you do not have a resolver setup file, create one.
2. Specify the percentage value and the required setting for the AUTOQUIESCE operand on the UNRESPONSIVETHRESHOLD statement in the resolver setup

file. The following guidelines apply when you modify the value for the UNRESPONSIVETHRESHOLD statement:

- If you change the value to 0, the function is disabled, all accumulated statistics are deleted, and any unresponsive name server notification messages are cleared from the operator console.
- If you change the value to a nonzero value but you do not change the value of the AUTOQUIESCE operand, the new threshold percentage will be used when the next time interval ends.
  - If you do not specify the AUTOQUIESCE operand and the resolver was already monitoring name server responsiveness, existing statistics that were accumulated during the 1-minute interval are not affected and current EZZ9308E unresponsive name server messages remain on the operator console.
  - If you specify the AUTOQUIESCE operand and the resolver was already monitoring name server responsiveness, existing statistics that were accumulated during the 30-second interval are not affected and current EZZ9311E unresponsive name server messages remain on the operator console.
- If you change the value of the AUTOQUIESCE operand, all accumulated statistics are deleted and any unresponsive name server messages (EZZ9308E or EZZ9311E) are cleared from the operator console. The resolver uses the specified threshold percentage when the next time interval (5-minutes or 30-seconds) ends.

3. Issue the MODIFY RESOLVER,REFRESH,SETUP=*setup_filename* command to cause the resolver to use the new threshold value, the AUTOQUIESCE operand setting, or both.

For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands .

Examples

To specify a 50% threshold value and cause the resolver to continue sending DNS queries generated by an application to an unresponsive name server, specify the following value:

```
UNRESPONSIVETHRESHOLD(50)
```

To specify a 100% threshold value and cause the resolver to stop sending DNS queries generated by an application to an unresponsive name server, specify the following values:

```
UNRESPONSIVETHRESHOLD(100,AUTOQUIESCE)
```

; and # statements

Use ; or # to indicate comments. Any data after the ; or # character is treated as a comment.

Configuration statements in TCPIP.DATA

The TCPIP.DATA configuration statements are summarized in Table 8.

Table 8. Summary of TCPIP.DATA configuration statements

Statement	Description	See
ALWAYSWTO	Issue WTO messages for servers.	"ALWAYSWTO statement" on page 373
DATASETPREFIX	Set the high-level qualifier for dynamic allocation of data sets.	"DATASETPREFIX statement" on page 374
DOMAINORIGIN or DOMAIN (see note)	Specify the domain origin that is appended to the host name to form the fully qualified domain name of a host.	"DOMAINORIGIN statement" on page 374
HOSTNAME	Specify the TCP host name of the z/OS communication server.	"HOSTNAME statement" on page 376
LOADDBCSTABLES	Indicate to FTP which DBCS translation tables can be loaded.	"LOADDBCSTABLES statement" on page 377
LOOKUP	Specify the order in which the DNS and the local host file are to be used for name resolution.	"LOOKUP statement" on page 378
MESSAGECASE	Specify case translation for the FTP server and osnmpd.	"MESSAGECASE statement" on page 379
NOCACHE	Specify that the resolver should not use the system-wide cache for any queries associated with applications that use this TCPIP.DATA file. The cache is bypassed for any queries from a DNS server and results obtained from a DNS server are not updated in the cache.	"NOCACHE statement" on page 381
NSINTERADDR or NAMESERVER **Note:** A synonym that provides common statements regardless of whether the statements are defined in an MVS data set or z/OS UNIX file.	Define the IP address of a name server. The IP address can be either IPv4 or IPv6.	"NSINTERADDR statement" on page 381
NSPORTADDR	Specify the name server port number.	"NSPORTADDR statement" on page 384
OPTIONS	Specify if resolver debug messages should be issued and the number of periods (.) that need to be contained in a domain name for it to be considered a fully qualified domain name.	"OPTIONS statement" on page 385
RESOLVERTIMEOUT	Specify how long the resolver waits for a response while trying to communicate with the name server.	"RESOLVERTIMEOUT statement" on page 387
RESOLVERUDPRETRIES	Specify how many times the resolver tries to connect to the name server when using UDP datagrams.	"RESOLVERUDPRETRIES statement" on page 389
RESOLVEVIA	Specify the protocol used by the resolver to communicate with the name server.	"RESOLVEVIA statement" on page 390
SEARCH	Specify the list of domain names that are appended, in the order listed, to the host name to form the fully qualified domain name of a host.	"SEARCH statement" on page 391

Table 8. Summary of TCPIP.DATA configuration statements (continued)

Statement	Description	See
SOCKDEBUG	Turn on tracing of TCP/IP socket library calls.	"SOCKDEBUG statement" on page 392
SOCKNOTESTSTOR	Stop checking of TCP/IP socket calls for storage access errors on the parameters to the call.	"SOCKNOTESTSTOR statement" on page 393
SOCKTESTSTOR	Enable checking of TCP/IP socket calls for storage access errors on the parameters to the call.	"SOCKTESTSTOR statement" on page 393
SORTLIST	Specify the ordered list of network numbers (subnets or networks) for the resolver to prefer if it receives multiple addresses as the result of a name query.	"SORTLIST statement" on page 394
TCPIPJOBNAME or TCPIPUSERID (see note)	Specify the member name of the cataloged procedure used to start the TCPIP address space.	"TCPIPJOBNAME statement" on page 395
TRACE RESOLVER	Trace all queries to and responses from the name server.	"TRACE RESOLVER statement" on page 396
TRACE SOCKET	Trace TCP/IP C socket library calls.	"TRACE SOCKET statement" on page 397
; or #	Use either character to indicate a comment.	"; and # statements" on page 368

Rule: If any TCPIP.DATA statement is in the GLOBALTCPIPDATA file, it is always used, regardless of what is found in any subsequent TCPIP.DATA statements from the search list.

system_name considerations

The *system_name* parameter on the statements is derived from the MVS system name. If you have configured VMCF and TNF as non-restartable subsystems, the system name is specified in the IEFSSNxx member of PARMLIB. If you have configured VMCF and TNF as restartable subsystems, the system name is obtained from the value of the P= parameter of the EZAZSSI started procedure. See "Step 3: Configure VMCF and TNF" in z/OS Communications Server: IP Configuration Guide for information about starting VMCF.

sysname above is the name specified by the IEASYSxx parmlib member's SYSNAME= parameter value. For more information about the SYS1.PARMLIB member definitions, see z/OS MVS Initialization and Tuning Guide.

For more information about using the *system_name* parameter, see the TCPIP.DATA statements customization information in z/OS Communications Server: IP Configuration Guide

For SMTP usage, use the NJENODENAME statement in the SMTP configuration data set to specify the JES nodename for mail delivery on the NJE network.

Dynamically changing TCPIP.DATA statements

You can use the MODIFY RESOLVER,REFRESH command to change some of the TCPIP.DATA statements being used by a long-running TCP/IP application (for example, a server application). To do this, follow either of the following procedures.

Restrictions: You cannot change the TCPIP.DATA statement values for the following subset of TCP/IP provided applications:

- SMTP server
- DNS utility z/OS UNIX commands (nslookup, onslookup and dig)
- Any application program that uses the Language Environment C/C++ res_ API facilities and changed the updated TCPIP.DATA statement

Table 9 lists the TCPIP.DATA statements and whether each statement can be dynamically changed (refreshed). For more information about modifying statements, see z/OS Communications Server: IP System Administrator's Commands and to z/OS Communications Server: IP Configuration Guide for information about configuring resolvers.

Table 9. Refreshable TCPIP.DATA

TCPIP.DATA statement	Refreshable
ALWAYSWTO	No
DATASETPREFIX	No
DOMAINORIGIN or DOMAIN	Yes
HOSTNAME	No
LOADDBCSTABLES	No
LOOKUP	Yes
MESSAGECASE	No
NOCACHE	Yes
NSINTERADDR or NAMESERVER	Yes
NSPORTADDR	Yes
RESOLVEVIA	Yes
RESOLVERTIMEOUT	Yes
RESOLVERUDPRETRIES	Yes
SEARCH	Yes
SOCKDEBUG	No
SOCKNOTESTSTOR	No
SOCKTESTSTOR	No
SORTLIST	Yes
TCPIPJOBNAME or TCPIPUSERID	No
TRACE RESOLVER	Yes
TRACE SOCKET	No
OPTIONS DEBUG	Yes
OPTIONS NDOTS	Yes
; or # (COMMENT)	NA

Steps for dynamically changing TCPIP.DATA statements without using GLOBALTCPIPDATA:

This topic describes the steps of dynamically changing the TCPIP.DATA statements, but not using GLOBALTCPIPDATA.

Procedure

Perform the following steps to dynamically change TCPIP.DATA statements without using GLOBALTCPIPDATA.

1. Change the MVS data set or z/OS UNIX file currently being used for TCPIP.DATA statements to the new values.

2. To use the changed values, issue the MODIFY RESOLVER,REFRESH command. When application programs that are configured to use the TCPIP.DATA file make their next resolver socket call (for example, gethostbyaddr or gethostbyname), the new values are used.

Step for dynamically changing TCPIP.DATA statements using GLOBALTCPIPDATA:

This topic describes the steps of dynamically changing the TCPIP.DATA statement by using GLOBALTCPIPDATA.

Procedure

Perform the following step to dynamically change TCPIP.DATA statements using GLOBALTCPIPDATA.

Use the preceding procedure to change the GLOBALTCPIPDATA file.

Steps for creating a new GLOBALTCPIPDATA data set or file

Alternatively, you could create a new GLOBALTCPIPDATA data set or file.

Procedure

Perform the following steps to create a new GLOBALTCPIPDATA data set or file.

1. Create a new resolver setup file in which the GLOBALTCPIPDATA statement points to the new TCPIP.DATA file.
2. To use the changed values, issue the MODIFY RESOLVER,REFRESH,SETUP= command specifying the new resolver setup name. When application programs make their next resolver socket call (for example, gethostbyaddr or gethostbyname), the new values are used.

Determining which TCPIP.DATA statements are being used

Use the Trace Resolver facility to determine which TCPIP.DATA values the resolver is using and where they were read from. See z/OS Communications Server: IP Diagnosis Guide for information about how to dynamically start the trace. After the trace is active, issue use the Netstat HOME/-h command to display the values. Issuing a ping of a host name from TSO and from the z/OS UNIX shell also shows activity to any DNSs that might be configured.

Syntax conventions for TCPIP.DATA configuration statements

Observe the following syntax conventions for TCPIP.DATA statements:

- A data set containing TCPIP.DATA statements must be fixed or fixed block with a suggested logic record length (LRECL) in the range 80 - 256. The data set should not contain line numbers, because the line numbers are treated as parameter values for statements that allow multiple parameters.
- A z/OS UNIX file containing TCPIP.DATA statements can have a maximum line length of 256.
- Only one statement is allowed on each line.
- A statement can start in any position on a line.
- Statements are not case sensitive.
- Statements can be preceded by an optional *system_name*.

- Static system symbols can be used in statements.
- A blank line is treated as a comment.
- For statements with a single parameter value or no parameter value, a blank after the value ends the statement. Anything on the line following the blank is treated as a comment.
- For statements accepting multiple parameter values (for example, SEARCH, LOOKUP, NSINTERADDR/NAMESERVER, SORTLIST, OPTIONS, and LOADDBCSTABLES), at least one blank followed by either a semicolon (;), or # character must precede any comments.
- If the same statement is encountered multiple times within a single TCPIP.DATA specification, the last statement takes effect. See the SEARCH, SORTLIST, NSINTERADDR/NAMESERVER, OPTIONS, and LOADDBCSTABLES statements for their unique processing of multiple statements.
- When Trace Resolver is in effect a warning message is written for any error in the specification of a statement or its parameters. The message is written to the specified Trace Resolver output location. Processing continues with the next line.
- Allocation errors (including volume offline conditions) cause the resolver service being requested to continue to be processed, but processing of TCPIP.DATA statements stops. Any already processed statements are used (for example, GLOBALTCPIPDATA statements). Defaults are assigned to any statements not specified.

  If an allocation error occurs when trying to use TCPIP.DATA statements, a message is issued to the Joblog/STDOUT. If Trace Resolver is in effect, a message is also written to the specified Trace Resolver output location.

ALWAYSWTO statement

Some TCP/IP servers, such as SMTP, SNMPQE, LPD, and Miscellaneous server, can use the ALWAYSWTO statement to issue all of their messages as Write To Operator (WTO) messages. This is in addition to their messages being sent to the server's MVS joblog output. Omitting the ALWAYSWTO statement causes the server messages to be sent only to the server's MVS job output.

Guideline: Do not specify ALWAYSWTO YES unless requested by IBM service. Specifying YES can generate a large volume of system operator console messages.

Syntax

```
►►──┬──────────────┬──ALWAYSWTO──┬─NO──┬──────────────────────────────────►◄
    └─system_name:─┘             └─YES─┘
```

Parameters

system_name:
: The name of the system to which this statement applies. See “system_name considerations” on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

YES
: Indicates that server messages are to be displayed on the console.

NO Indicates that server messages only go to the MVS output.

DATASETPREFIX statement

Use the DATASETPREFIX statement to set the high-level qualifier for the dynamic allocation of data sets in TCP/IP.

Syntax

```
►►──┬──────────────┬──DATASETPREFIX ──dsprefix──────────────────────────►◄
    └─system_name:─┘
```

Parameters

system_name:
: The name of the system to which this statement applies. See “system_name considerations” on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

dsprefix
: The prefix to use as the high-level qualifier for the dynamic allocation of data sets. The default high-level qualifier distributed with the system is TCPIP.

  Guidelines: The values for the parameter must conform to the following rules:
  - A maximum of 26 characters.
  - Must contain one or more tokens separated by a period.
  - Each token must be in the range 1 - 8 characters in length.
  - Each token must start with a letter or character ($, @, or #).
  - Remaining characters in each token must be a letter, number, or character (-, $, @, or #).
  - The last character of the data set prefix must not be a period.

Examples

Code the following example to set the data set prefix for client and server usage to TCPIP.V1R6:

```
DATASETPREFIX TCPIP.V1R6
```

Usage notes

The DATASETPREFIX in TCPIP.DATA is used by clients and servers except the TCPIP address space.

DOMAIN statement

The DOMAIN statement is functionally equivalent to the DOMAINORIGIN statement. See “DOMAINORIGIN statement.”

DOMAINORIGIN statement

Use the DOMAINORIGIN statement to specify the domain origin that is appended to the host name to form the fully qualified domain name of a host.

Syntax

```
►►─┬──────────────┬─DOMAINORIGIN──origin──────────────────────────────►◄
   └─system_name:─┘
```

Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

origin
: The domain origin is appended to the host name. This name usually has imbedded dots.

  Guidelines: The values for the domain name must conform to the following rules:

  - Maximum of 249 characters.
  - Must contain one or more tokens separated by a period.
  - Each token must be at least one character.
  - Each token must start with a letter or number.
  - Remaining characters in each token must be a letter, number, or hyphen.
  - The length of the host name plus the length of the domain name must be less than or equal to 254.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

This example appends the domain origin of BOBS.YOUR.UNCLE to the host name:

```
DOMAINORIGIN  BOBS.YOUR.UNCLE
```

Usage notes

- No case translation is performed on the domain origin.
- If the resolver is passed a host name that does not contain any dots (in dotted decimal notation), the domain origin is appended to the host name. If the host name passed to the resolver contains dots, the value of the OPTIONS NDOTS:n statement influences how the DOMAINORIGIN value is used. See "OPTIONS statement" on page 385.
- The DOMAINORIGIN configuration statement must be customized at each site.
- Additionally, the domain origin can be set from the z/OS shell environment by exporting the LOCALDOMAIN environment variable.

  ```
  ►►──export──LOCALDOMAIN=origin────────────────────────────────────────►◄
  ```

  The setting of the LOCALDOMAIN as an environment variable overrides any setting for DOMAIN, DOMAINORIGIN, or SEARCH found in TCPIP.DATA.

HOSTNAME statement

Use the HOSTNAME statement to specify the TCP host name of this z/OS Communications Server server.

Syntax

```
►►──┬──────────────┬──HOSTNAME──host_name──────────────────────────────►◄
    └─system_name:─┘
```

Parameters

*system_name***:**
The name of the system to which this statement applies. See “system_name considerations” on page 370 for a complete description of this parameter.

Requirement: The colon is required.

host_name
The host name. If not specified or if not valid, the value is determined as follows:

- The system name specified in the IEFSSNxx PARMLIB member or on the P parameter of the EZASSI started procedure used for restartable VMCF. See z/OS Communications Server: IP Configuration Guide for details about configuring VMCF.
- If VMCF was not active at the time the TCP/IP stack was started, the CVTSNAME value (this is the SYSNAME=value in the IEASYSxx PARMLIB member A).

If the host name came from TCPIP.DATA, it is in the message case it was specified in on the HOSTNAME statement. For VMCF or CVTSNAME, the name is upper case.

The TCP/IP stack's configuration function uses the z/OS UNIX search order to locate the TCPIP.DATA HOSTNAME statement to determine the stack's host name. See search orders used in the z/OS UNIX environment in the z/OS Communications Server: IP Configuration Guide for a description of this search order. This host name value is the value that is returned on gethostname socket function calls processed by the stack. If the HOSTNAME statement is changed, TCP/IP needs to be restarted to pick up this change.

Guidelines: The values for the host name must conform to the following rules:

- Maximum of 63 characters.
- Must contain one or more tokens separated by a period.
- Each token must be at least one character and less than 64 characters.
- Each token must start with a letter or number.
- Remaining characters in each token must be a letter, number, or hyphen.

Examples

The TCPIP.DATA data set is shared between two systems, MVSMFG4 and MVSADM1. The HOSTNAME statements define the host name on each system.

```
MVSMFG4: HOSTNAME MVSMFG4
MVSADM1: HOSTNAME MVSADM1
```

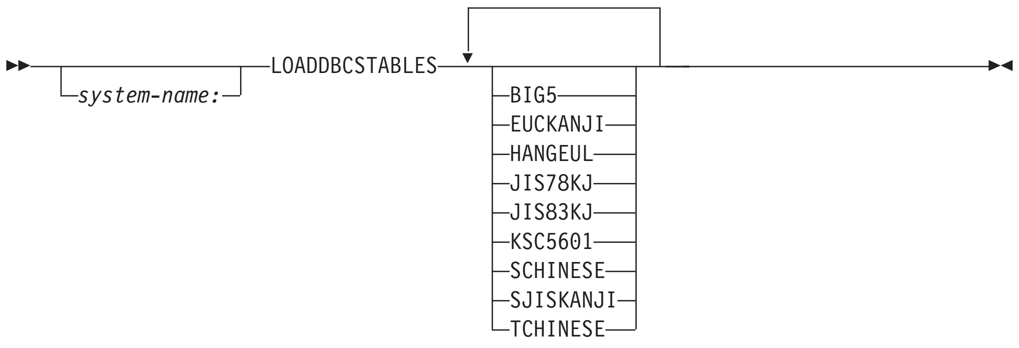
Usage notes

- No case translation is performed on the host name.
- See z/OS Communications Server: IP Configuration Guide for descriptions of local host tables.

LOADDBCSTABLES statement

Use the LOADDBCSTABLES statement to indicate to the FTP server and client which DBCS translation tables can be loaded.

Syntax



Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

BIG5
Indicates to the FTP server and client that the BIG5 DBCS translation table should be loaded from the TCPCHBIN binary translate table data set.

EUCKANJI
Indicates to the FTP server and client that the Extended UNIX Code Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table data set.

HANGEUL
Indicates to the FTP server and client that the Hangeul DBCS translation table should be loaded from the TCPHGBIN binary translate table data set.

JIS78KJ
Indicates to the FTP server and client that the JIS 1978 Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table data set.

JIS83KJ
Indicates to the FTP server and client that the JIS 1983 Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table data set.

KSC5601
Indicates to the FTP server and client that the Korean Standard Code KSC-5601 DBCS translation table should be loaded from the TCPHGBIN binary translate table data set.

SCHINESE
Indicates to the FTP server and client that the Simplified Chinese DBCS translation table should be loaded from the TCPSCBIN binary translate table data set.

SJISKANJI
Indicates to the FTP server and client that the Shift JIS Kanji DBCS translation table should be loaded from the TCPKJBIN binary translate table data set.

TCHINESE
Indicates to the FTP server and client that the Traditional Chinese (5550) DBCS translation table should be loaded from the TCPCHBIN binary translate table data set.

Examples

Load the Korean Standard Code KSC-5601 and the Traditional Chinese (5550) DBCS translation tables:

```
LOADDBCSTABLES KSC5601 TCHINESE
```

Usage notes

- You can select any or all of the translation tables or specify none. However, additional virtual storage might be required by the FTP server and client when a large number of translation tables are loaded at the same time.
- All the parameters must fit on one line. You can repeat the LOADDBCSTABLES statement as necessary to specify additional tables to be loaded.
- If the LOADDBCSTABLES parameter is not specified, is specified incorrectly, or if TCPIP.DATA is not accessible, then no DBCS translation tables are Reloaded, and the corresponding FTP server and client DBCS transfer types are unavailable.
- The IBMKANJI transfer type does not require any translation table to be loaded.
- If the same table name is specified more than one time, the subsequent specifications are ignored.

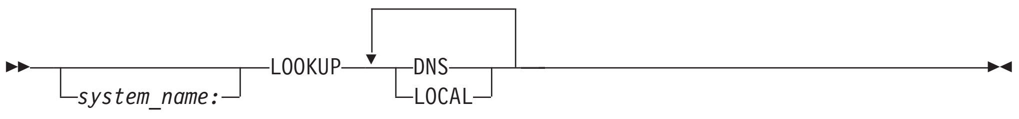
Related topics

See z/OS Communications Server: IP Configuration Guide for more information.

LOOKUP statement

Use the LOOKUP statement to specify the order in which the DNS or local host tables are to be used for name resolution.

Syntax



Parameters

system_name:
The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

`DNS`
: The domain name servers specified by the NSINTERADDR and NAMESERVER statements are used for name resolution. When system-wide caching is active, this processing includes querying the resolver cache first for entries provided by these name servers on previous name resolution attempts, and only if that query fails, querying the domain name servers.

  For more resolver information see z/OS Communications Server: IP Configuration Guide for more details.

`LOCAL`
: The local host tables (for example, etc/hosts, HOSTS.SITEINFO or HOSTS.ADDRINFO) are used for name resolution. See z/OS Communications Server: IP Configuration Guide for information about determining which local host tables are used.

Statement dependency

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

In the following example, only the local host tables are used:

```
LOOKUP LOCAL
```

In the following example, the local host tables are used first. If the resource name is not resolved, then the resolver cache (if a cache is being used) is used next. If there is still no resolution, then the name servers are queried directly.

```
LOOKUP LOCAL DNS
```

In the following example, the resolver cache (if a cache is being used) is used first. If the resource name is not resolved, then the name servers are queried directly. If there is still no resolution, then the local host tables are used next.

```
LOOKUP DNS LOCAL
```

Usage notes

- If a LOOKUP statement is not specified, the resolver cache (if a cache is being used) is queried first. If the cache query is unsuccessful, the domain name servers are queried next, and if the resolution request is not successful, the local host file, if it exists, is used.
- If an incorrect parameter value is specified, the entire LOOKUP statement is ignored.
- The last syntactically correct LOOKUP statement is used.

MESSAGECASE statement

Use the MESSAGECASE statement to specify whether to convert output into uppercase for the FTP server and some TSO commands.

Syntax

```
                                  ┌─MIXED─┐
►►──┬──────────────┬──MESSAGECASE──┴─UPPER─┴──────────────────────────────►◄
    └─system-name:─┘
```

Parameters

`system_name`**:**
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

`MIXED`
: Indicates that output should be displayed in mixed case. This is the default.

`UPPER`
: Indicates that output should be displayed in uppercase.

Examples

To display all messages to the MVSTEST system in uppercase, use the following code:

```
MVSTEST: MESSAGECASE UPPER
```

Usage notes

- If you specify MIXED, no case conversion is performed on output.
- If the MESSAGECASE statement is not specified, is specified incorrectly, if MIXED or UPPER is not specified, or if TCPIP.DATA is not accessible, then mixed case output is displayed.
- All Writer To Operator (WTO) messages issued by the TCPIP stack are displayed in uppercase and are not affected by the MESSAGECASE value.
- Additionally, the MESSAGECASE statement can be set from the z/OS shell environment by exporting the MESSAGECASE environment variable. The MESSAGECASE environment variable is not supported by all functions. This is shown in the following example:

  ```
                              ┌─MIXED─┐
  ►►──export──MESSAGECASE=────┴─UPPER─┴──────────────────────────────►◄
  ```

  The setting of the MESSAGECASE as an environment variable overrides any setting found in TCPIP.DATA. If MESSAGECASE is not defined as an environment variable or as a statement in TCPIP.DATA, the WTO message remains in mixed case.

NAMESERVER statement

The NAMESERVER statement is functionally equivalent to the NSINTERADDR statement. See "NSINTERADDR statement" on page 381.

NOCACHE statement

Use the NOCACHE statement to indicate that results from application queries that are associated with this TCPIP.DATA file are not used to populate the system cache, nor are the contents of the system cache used to generate results to application queries.

For more information about resolvers see z/OS Communications Server: IP Configuration Guide for more details.

Syntax

```
►►──┬──────────────┬──NOCACHE──────────────────────────────────────────►◄
    └─system_name:─┘
```

Parameters

system_name:
The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

Steps for modifying

You can use the MODIFY command to change whether or not system-caching functions are used. If the NOCACHE statement is not in TCPIP.DATA, the current system-wide settings for use of the caching function should be used.

For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

NSINTERADDR statement

Use the NSINTERADDR statement to define the IP address of a name server. The IP address can be either an IPv4 address or an IPv6 address.

See z/OS Communications Server: IP Configuration Guide for implications on coding an IPv6 address on this statement.

Syntax

```
                                   ┌────────────────┐
                                   ▼                │
►►──┬──────────────┬──NSINTERADDR────internet_addr──┴──────────────────►◄
    └─system_name:─┘
```

Parameters

system_name:
The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

internet_addr
The IP address of a name server.

Guidelines:

- You must code the values for a name server IPv4 address in dotted decimal format. The following restrictions apply:
  - You cannot specify the following IPv4 addresses as a valid name server IPv4 address:
    - IPv4 unspecified address (0.0.0.0)
    - IPv4 broadcast address
    - IPv4 multicast address
  - You cannot specify IPv4 subnet length information as part of the IPv4 address.
- You must code the values for a name server IPv6 address in colon hexadecimal format. You can specify the IPv6 addresses in upper case, lower case, or mixed case formats. The following restrictions apply:
  - You cannot specify the following IPv6 addresses as a valid name server IPv6 address:
    - IPv6 unspecified address (::)
    - IPv6 multicast address
    - IPv4-mapped IPv6 address
    - IPv6 address with the reserved prefix ::/96
  - You cannot specify scope information as part of the IPv6 address.
  - You cannot specify IPv6 prefix length information as part of the IPv6 address.

You can specify multiple IP addresses on a single NSINTERADDR statement. You can specify all IPv4 addresses, all IPv6 addresses, or a mixture of IPv4 and IPv6 addresses, in any order, on a single statement.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

To specify the IP address of the name server to be 14.13.12.11, use the following code:

```
NSINTERADDR  14.13.12.11
```

To specify the IP address of the three name servers to be 14.13.12.11, 9.9.9.9, and 6.7.8.9, use the following code:

```
NSINTERADDR  14.13.12.11  9.9.9.9  6.7.8.9
```

An equivalent specification is:

```
NSINTERADDR  14.13.12.11
NSINTERADDR  9.9.9.9
NSINTERADDR  6.7.8.9
```

To specify the IP address of the name server to be 2001:0000:0000:0000:000E:000D:000C:000B, code any of these equivalent definitions:

```
NSINTERADDR  2001:0000:0000:0000:000E:000D:000C:000B
NSINTERADDR  2001::e:d:c:b
NSINTERADDR  2001:0:0:0:E:d:c:B
```

To specify the IP address of the three name servers to be 2001::e:d:c:b, 2001::9:9:9:9, and 2001:6:7:8:9, use the following code:

```
NSINTERADDR  2001::e:d:c:b  2001::9:9:9:9  2001::6:7:8:9
```

An equivalent specification is:

```
NSINTERADDR  2001::e:d:c:b
NSINTERADDR  2001::9:9:9:9
NSINTERADDR  2001::6:7:8:9
```

To specify a mixture of IPv4 and IPv6 name server IP addresses, for example ::1, 14.13.12.11, 9.9.9.9, and 2001:6:7:8:9, use the following code:

```
NSINTERADDR  ::1  14.13.12.11  9.9.9.9  2001::6:7:8:9
```

An equivalent specification is:

```
NSINTERADDR  ::1
NSINTERADDR  14.13.12.11
NSINTERADDR  9.9.9.9
NSINTERADDR  2001::6:7:8:9
```

Usage notes

- Up to 16 name server IP addresses can be specified. Any IP addresses beyond 16 are ignored.
- Connections to the name servers are attempted in the order they appear in the TCPIP.DATA data set.

  If network DNS response message sizes tend to be larger than 512 bytes, put name servers that support Extension Mechanisms for DNS (EDNS0) before name servers that do not support EDNS0. The z/OS resolver supports UDP message sizes as large as 3072 bytes when communicating with name servers that support EDNS0. Ordering the name servers in terms of EDNS0 support can potentially avoid the use of more expensive TCP protocols when processing large DNS response messages.
- If resolver caching is in effect, the resolver cache is searched first before connections to any of the name servers are attempted. If valid, non-expired response data for the target resource has been received and cached from any name server in the list, the resolver uses that data and does not send queries to any name servers. If response data for the target resource has been received and cached from more than one name server in the list, the resolver chooses the response data to be returned based on the order of the name servers in the TCPIP.DATA data set.
- If no NSINTERADDR statements are coded, the resolver does not attempt to use a name server. Instead, the resolver uses the local host tables as described in z/OS Communications Server: IP Configuration Guide to attempt to resolve the name or IP address.
- You can specify the same IP address multiple times.
- If you specify multiple name server IP addresses on a single NSINTERADDR statement, if any IP address is not an acceptable IP address, all IP addresses on the NSINTERADDR statement are ignored.
- After the resolver has successfully contacted a name server, it stops without contacting the remaining name servers for that query. Name servers beyond the first in the list are used only if the name server currently being contacted is down, or unreachable through the network.

- When the AUTOQUIESCE operand is specified on the UNRESPONSIVETHRESHOLD resolver setup statement, the resolver attempts to contact a name server for DNS queries generated by an application, based on whether the name servers specified on the NSINTERADDR statements are responsive.
  - If all name servers specified on the NSINTERADDR statements are responsive, or if all name servers are unresponsive, the resolver contacts all name servers in the list until a name server is successfully contacted or all attempts have failed.
  - If only a subset of name servers specified on the NSINTERADDR statements are responsive, the resolver attempts to contact only those name servers that are responsive, and does not send queries that are generated by an application to unresponsive name servers.

  See the "UNRESPONSIVETHRESHOLD statement" on page 366 for more information about when the resolver considers a name server to be unresponsive.
- RESOLVERUDPRETRIES indicates the maximum number of times an attempt is made to reach a given name server if a response is not received within the current timeout interval. RESOLVERUDPRETRIES is applicable only if RESOLVEVIA UDP is coded or used by default.

  Tip: RESOLVERTIMEOUT is the parameter used for the timeout value.

NSPORTADDR statement

Use the NSPORTADDR statement to specify the name server port number.

Syntax

```
                            ┌─NSPORTADDR 53──────────┐
►►─┬───────────────┬────────┼────────────────────────┼──────────────────────►◄
   └─system_name:──┘        └─NSPORTADDR──nsportaddr─┘
```

Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

nsportaddr
: The name server port number. The default is port 53.

  Guidelines: The values for the name server port must conform to the following rules:
  - Must be a single number.
  - The number must be between one and five digits.
  - The number cannot exceed 65 535.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

To specify the foreign port of the name server to be 55, use the following code:

```
NSPORTADDR 55
```

OPTIONS statement

Use the OPTIONS statement to specify:

- Whether or not resolver debug messages should be issued
- The number of periods (.) that need to be contained in a domain name for it to be considered a fully qualified domain name

Guideline: The NDOTS and DEBUG options are independent; setting one of them does not imply a setting for the other.

Syntax

```
                                 ┌───────────┐
                                 ▼           │
►►──┬──────────────┬──OPTIONS────┬─────────┬─┴────────────────────────────────────►◄
    └─system-name:─┘             ├─DEBUG───┤
                                 └─NDOTS:n─┘
```

Parameters

***system_name*:**
The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

DEBUG
Specifying DEBUG is equivalent to the Trace Resolver statement. Debugging messages from the resolver are generated.

If OPTIONS DEBUG is anywhere in the TCPIP.DATA file, then tracing is on. It should be the first statement in the TCPIP.DATA statements to get the maximum trace output. The initial default setting is for no debug messages to be specified. Do not specify OPTIONS DEBUG in the GLOBALTCPIPDATA file.

NDOTS:*n*
Specifies that for a domain name that contains *n* or more periods (.), the resolver should try to look up the name *as is* before applying the DOMAINORIGIN or SEARCH statement settings.

Requirement: The colon is required.

A maximum of 15 is allowed for *n*. Any value for *n* not in the range 1 - 15 results in *n* being set to 1. Not specifying the NDOTS:*n* parameter results in the current setting remaining in effect (if no value has yet been specified on any previous OPTIONS statements, then NDOTS:1 is the setting).

Use care when setting n greater than 1. For example, consider the following:

- If NDOTS:2 was specified and the DOMAINORIGIN statement had mit.edu specified, the following results would be observed:

  - A user enters `ftp prep.ai`. Resolution of domain name prep.ai.mit.edu would be tried. If that fails resolution, then the name prep.ai would be tried.
  - A user enters `ftp prep.ai.mit`. The domain name prep.ai.mit would try to be resolved. If that fails resolution, then the name prep.ai.mit.mit.edu would be tried.
  - A user enters `ftp prep`. The domain name prep.mit.edu would try to be resolved. If that fails resolution, then the name prep would be tried.
- If NDOTS:1 was specified and the SEARCH statement had ai.mit.edu and MIT.EDU specified, the following results would be observed:
  - A user enters `ftp prep.ai`. The domain name `prep.ai` would try to be resolved. If that fails resolution, then the name `prep.ai.ai.mit.edu` would be tried. If that fails resolution, then the name `prep.ai.MIT.EDU` would be tried.
  - A user enters `ftp prep`. The domain name `prep.ai.mit.edu` would try to be resolved. If that fails resolution, then the name `prep.MIT.EDU` would be tried. If that fails resolution, then the name `prep` would be tried.
- If the name specified by the user ends with a period (.), then both the NDOTS:*n* specification and the DOMAINORIGIN or SEARCH values are ignored. For example, a user enters `ftp prep.ai.`. The domain name `prep.ai.` would try to be resolved. If that fails, no other name is tried.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

The following statement sets NDOTS to 2 and also requests resolver debug messages:

```
OPTIONS NDOTS:2 DEBUG
```

The following statement requests resolver debug messages and by default set NDOTS to 1:

```
OPTIONS DEBUG
```

The following set of statements in a single TCPIP.DATA file sets NDOTS to 3 and also request resolver debug messages:

```
OPTIONS NDOTS:2 DEBUG
OPTIONS NDOTS:3
```

The following set of statements in a single TCPIP.DATA file would set NDOTS to 3 and also request resolver debug messages.

```
OPTIONS NDOTS:2
OPTIONS NDOTS:3 DEBUG
OPTIONS
```

Usage notes

- If the OPTIONS statement is not specified or specified without a NDOTS:*n* parameter (for example, OPTIONS specified only with the DEBUG parameter), a value of :1 is assigned. Do not specify the OPTIONS DEBUG parameter in the GLOBALTCPIPDATA file.

Guideline: This assumes only one OPTIONS statement in the TCPIP.DATA file.

- If multiple OPTIONS NDOTS:*n* statements are encountered in a single TCPIP.DATA file, the last statement takes effect.
- If an OPTIONS statement without the DEBUG parameter is specified, the previous debug setting stays in effect. The default setting is for no debug messages to be specified.

Related topics

- "DOMAINORIGIN statement" on page 374
- "SEARCH statement" on page 391
- "TRACE RESOLVER statement" on page 396
- z/OS Communications Server: IP Configuration Guide

RESOLVERTIMEOUT statement

Use the RESOLVERTIMEOUT statement to specify the amount of time the resolver waits for a response while trying to communicate with a name server when using UDP. See "RESOLVEVIA statement" on page 390.

Syntax

```
                                 ┌─RESOLVERTIMEOUT 5─────────────────┐
►►─┬───────────────┬─────────────┼───────────────────────────────────┼──────────────────────►◄
   └─system_name:──┘             └─RESOLVERTIMEOUT──time_out_value───┘
```

Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

time_out_value
: The time the resolver waits until a response is received from a name server. The time can be specified in whole seconds, milliseconds, or a combination of both. For example, the RESOLVERTIMEOUT value can be 11, .110, or 1.100.

  A *time_out_value* value that is less than 10 milliseconds is set to 10 milliseconds (0.010). For example, RESOLVERTIMEOUT 0.005 is processed as RESOLVERTIMEOUT 0.010.

  A *time_out_value* value of 0 is equivalent to RESOLVERTIMEOUT 1.

  Specifying more than three decimal positions is considered a parse error and is ignored. For example, RESOLVERTIMEOUT 0.0100 is a parse error and is not processed.

  The default timeout value is 5 seconds; the maximum timeout value is 2147483.647.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

Specify a 10 second *time_out_value* value:

```
RESOLVERTIMEOUT 10
```

Specify a half second *time_out_value* value:

```
RESOLVERTIMEOUT .5
```

Specify a 75 millisecond *time_out_value* value:

```
RESOLVERTIMEOUT .075
```

Specify a 3 and one half second *time_out_value* value:

```
RESOLVERTIMEOUT 3.50
```

Usage notes

The resolver uses the RESOLVERTIMEOUT value when it is waiting for a response to a resolver DNS polling query. The resolver sends resolver DNS polling queries to a name server when AUTOQUIESCE is specified on the UNRESPONSIVETHRESHOLD resolver setup statement and the name server is considered unresponsive. If the RESOLVERTIMEOUT value is changed using the MODIFY RESOLVER,REFRESH command, the new time value applies only to resolver DNS polling queries that are sent by the resolver after the MODIFY command is processed. See "UNRESPONSIVETHRESHOLD statement" on page 366 for more information about when the resolver considers a name server to be unresponsive.

Guideline: If you use the autonomic quiescing of unresponsive name servers function, you should specify a timeout value of 5 seconds or less.

The SMTP server and the BIND 9 DNS utilities provide their own resolver that supports RESOLVERTIMEOUT values in seconds. If a *time_out_value* of less than 1 second is specified, these resolvers use a one second timeout. For a *time_out_value* of seconds.milliseconds, the specified seconds are used as the timeout value.

Be careful when assigning a short *time_out_value*. A number too small can result in timeouts occurring even when the network or name server is available, but due to high usage volume, it cannot respond quickly. Review the RESOLVERUDPRETRIES statement to see if a higher value should be specified for the maximum number of tries the resolver can make when using a name server.

Tip: Timeout conditions can cause the z/OS resolver to mistakenly act as though the name server does not support Extension Mechanism for DNS (EDNS0). This can prevent the z/OS resolver from using EDNS0 when it could otherwise be used; this behavior can adversely affect performance. For more information about ENDS0 processing, see z/OS Communications Server: IP Configuration Guide.

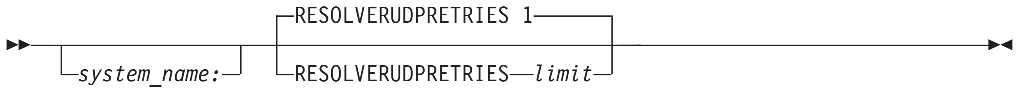
Guideline: The resolver uses the API services of z/OS Unix System Services to manage the *time_out_value* value. z/OS Unix System Services uses the following criteria for timer resolution, if the *time_out_value* is one of the following values:

- Less than 1 second, the timer resolution is set to the next microsecond.
- Greater than 1 second, the timer resolution is set to the next second.

RESOLVERUDPRETRIES statement

Use the RESOLVERUDPRETRIES statement to specify the number of times (including retries) the resolver should try to connect to the name server when using UDP datagrams.

Syntax



Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

limit
: The maximum number of times the resolver should try to connect to the name server. The default is 1; the maximum number can be 2 147 483 647.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

To specify 2 as the number of times the resolver tries to connect to the name server when using UDP datagrams, use the following code:

```
RESOLVERUDPRETRIES 2
```

Usage notes

- This statement applies only when using UDP datagrams. See "RESOLVEVIA statement" on page 390 for more information.
- The resolver attempts to contact each of the specified name servers before attempting any retries.
- The maximum amount of time for each UDP resolution is the product of the number of name servers (NSINTERADDR/NAMERSERVER statements) multiplied by the resolver timeout value (RESOLVERTIMEOUT statement) multiplied by the number of times to try the name servers (RESOLVERUDPRETRIES statement). This amount of time can occur for each domain name specified by the SEARCH statement. If a getaddrinfo API call is issued to request a query for both IPv4 and IPv6 addresses, the maximum amount of time can be doubled.
- A RESOLVERUDPRETRIES value of zero indicates that the resolver should not attempt to contact any name servers.
- Use the DIG command with the STATS option to determine how many attempts it takes for each DNS in the NSINTERADDR list to respond. Set RESOLVERUDPRETRIES to the number of attempts for the least responsive DNS in the list, and place the least responsive DNSs at the end of the list.

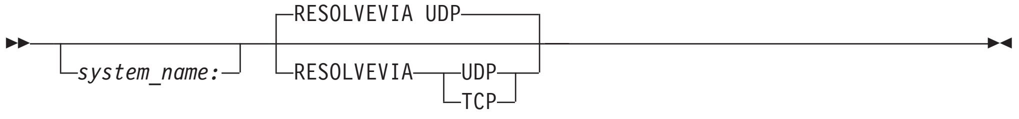
- If network DNS response message sizes tend to be larger than 512 bytes, put name servers that support Extension Mechanisms for DNS (EDNS0) before name servers that do not support EDNS0. The z/OS resolver supports UDP message sizes as large as 3 072 bytes when communicating with name servers that support EDNS0. Ordering the name servers in terms of EDNS0 support can potentially avoid the use of more expensive TCP protocols when processing large DNS response messages.

RESOLVEVIA statement

Use the RESOLVEVIA statement to specify the protocol used by the resolver to communicate with the name server.

Guideline: If you use the autonomic quiescing of unresponsive name servers function, you should use UDP as your protocol to communicate with the name server.

Syntax



Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

UDP
Specifies that the protocol is UDP. The default protocol is UDP.

TCP
Specifies that the protocol is TCP.

If anything other than UDP or TCP is specified, the default of UDP is used.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

To specify that the resolver is to communicate with the name server using TCP virtual circuits, code the following:

```
RESOLVEVIA TCP
```

Usage notes

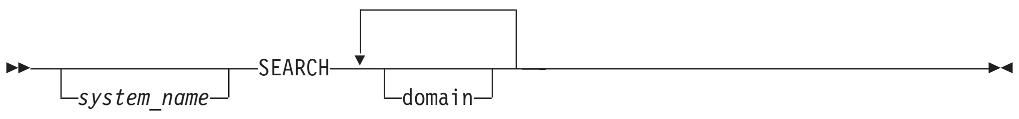
When RESOLVEVIA UDP is specified, the resolver primarily uses the UDP protocol but can switch to TCP protocols under certain conditions. The resolver is most likely to switch to TCP protocols when a DNS reply from the name server is truncated. Even though the resolver supports Extension Mechanisms for DNS0 (EDNS0) standards and accepts up to 3072 bytes of reply data, not all DNS name servers support EDNS0. Firewall settings along the path to the name server can also limit the number of bytes in a resolver UDP reply. To maximize the use of UDP protocols by the resolver, configure firewalls such that the TCP and UDP ports to and from the name servers are allowed to pass the larger EDNS0 packets.

For more information about EDNS0 processing, see z/OS Communications Server: IP Configuration Guide.

SEARCH statement

Use the SEARCH statement to specify the list of domain names that are appended, in the order listed, to the host name to form the fully qualified domain name of a host. A domain name is appended until either the list is exhausted or an IP address is determined. The domain names are appended for name server queries as well as for searching the local host tables.

Syntax



Parameters

system_name:
The name of the system to which this statement applies. See “system_name considerations” on page 370 for a complete description of this parameter.

Requirement: The colon is required.

domain
The domain name is appended to the host name. This name usually has imbedded dots.

For name query performance reasons, the first domain listed should be the most likely to respond to a name query. See “DOMAINORIGIN statement” on page 374 for the rules for the domain name values. See “RESOLVERTIMEOUT statement” on page 387 and “RESOLVERUDPRETRIES statement” on page 389 for details. No case translation is performed on the domain name.

Up to six names separated by at least one blank are allowed. If the domain names cannot fit on a single SEARCH statement, multiple SEARCH statements can be used. If more than six domain names are specified, only the first six are used. The first domain name specified is used as the value for DOMAINORIGIN/DOMAIN. If both the SEARCH and DOMAINORIGIN/ DOMAIN statements are present, the one that appears last is used. Encountering a DOMAINORIGIN/DOMAIN statement after SEARCH statements results in the DOMAINORIGIN's value as the only domain name.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

The following example would establish a search list:

```
SEARCH  raleigh.ibm.com   US.IBM.COM  ibm.com
```

An equivalent specification is:

```
SEARCH raleigh.ibm.com
SEARCH US.IBM.COM
SEARCH ibm.com
```

If a user entered FTP RALVM12 and assuming that OPTIONS NDOTS:*n* (see "OPTIONS statement" on page 385) was specified such that the SEARCH domains should be appended, the following order of name queries would be done in sequence by the resolver until either an answer was found, or the list was exhausted:

1. RALVM12.raleigh.ibm.com
2. RALVM12.US.IBM.COM
3. RALVM12.ibm.com

Related topics

- "DOMAINORIGIN statement" on page 374
- "OPTIONS statement" on page 385
- z/OS Communications Server: IP Configuration Guide

SOCKDEBUG statement

Use the SOCKDEBUG statement to turn on the tracing of TCP/IP socket library calls. This statement produces trace messages only for sockets using the TCP/IP C sockets or TCP/IP REXX sockets application programming interfaces.

Syntax

```
►►──┬───────────────┬──SOCKDEBUG──────────────────────────────────────►◄
    └─system_name:──┘
```

Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

Usage notes

This statement works for all TCP/IP C sockets across the system the way sock_debug() works for a specific socket application.

Related topics

See z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference for more information about sockets.

SOCKNOTESTSTOR statement

Use the SOCKNOTESTSTOR statement to stop checking of TCP/IP C sockets socket calls for storage access errors on the parameters to the call.

Syntax

```
►►──┬──────────────┬──SOCKNOTESTSTOR──────────────────────────────────►◄
    └─system_name:─┘
```

Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

Usage notes

- This statement improves response time.
- This statement is in effect unless SOCKTESTOR is specified.
- This statement works for all TCP/IP C sockets across the system the way sock_do_test_stor() works for a specific socket application.

Related topics

See z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference for more information about sockets.

SOCKTESTSTOR statement

Use the SOCKTESTSTOR statement to enable checking of TCP/IP C sockets socket calls for storage access errors on the parameters to the call.

Syntax

```
►►──┬──────────────┬──SOCKTESTSTOR────────────────────────────────────►◄
    └─system_name:─┘
```

Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

Usage notes

This statement works for all TCP/IP C sockets across the system the way sock_do_test_stor() works for a specific socket application.

Related topics

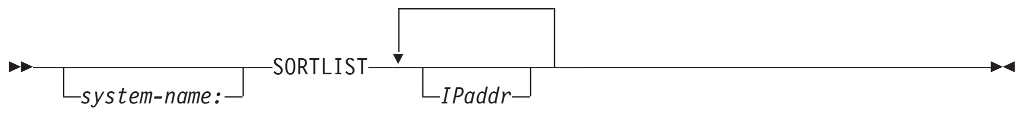
See z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference for more information about sockets.

SORTLIST statement

Use the SORTLIST statement to specify the ordered list of network numbers (subnets or networks) for the resolver to prefer if it receives multiple addresses as the result of a name query. This controls the list of addresses returned for a gethostbyname call. This is also used to sort the IPv4 addresses returned for a getaddrinfo call.

Restriction: This statement supports only IPv4 IP addresses.

Syntax



Parameters

system_name:
The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

IPaddr
The subnet or network address.

The specification of the address can be:

- network/subnet mask; for example 128.32.42.0/255.255.255.0 or 128.32.42.0/24

  The mask can be specified by a /xx. The number, denoted by xx, represents the number of significant bits in the mask, for example: /24=24 significant bits=11111111 11111111 11111111 00000000=255.255.255.0
- network; for example 128.32.0.0 or 9.0.0.0

  If no mask is specified then the following mask is used:
  - Class A network - 255.0.0.0
  - Class B network - 255.255.0.0
  - Class C network - 255.255.255.0
  - Class D or E network - 255.255.255.255

Guidelines: The values for the SORTLIST IP address must conform to the following rules:

- Must contain four tokens, each separated by a period.
- Each token must be between one and three characters.
- Each character in each token must be a number.
- Each token cannot exceed the 255.

The values for the SORTLIST subnet mask:

- The short format is of the form `x.x.x.x/y` where:

  - x.x.x.x is the IP address
  - y is an integer from 1 to 32 representing the number of bits for the mask
- The full format is of the form `x.x.x.x/y.y.y.y` where:
  - x.x.x.x is the IP address
  - y.y.y.y is the mask (same syntax checking as IP address)

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

In this example, assume that your host has multiple subnet interfaces, for example, 128.32.42 for FDDI and 128.32.1 for Ethernet.

If you want your applications to see the FDDI subnet address before any other interface address, code the SORTLIST statement as follows:

```
SORTLIST 128.32.42.0/24
```

If you want to ensure that FDDI is first and then any other Class B interface for 128.32, code the SORTLIST statement as follows:

```
SORTLIST 128.32.42.0/24 128.32.0.0
```

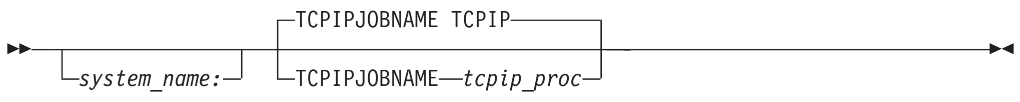
Usage notes

- A maximum of four IP addresses is allowed. If the IP addresses cannot fit on a single SORTLIST statement, multiple SORTLIST statements can be used. If more than four are specified, only the first four IP addresses are used.
- SORTLIST is supported only for GETHOSTBYNAME and GETADDRINFO calls that return IPv4 addresses, and is not used for NSLOOKUP or ONSLOOKUP.

TCPIPJOBNAME statement

Use the TCPIPJOBNAME statement to specify the member name of the procedure used to start the TCP/IP address space.

Syntax



Parameters

system_name:
: The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

  Requirement: The colon is required.

tcpip_proc
: The name of the member in the cataloged procedure library that is used to start the TCP/IP address space. In some cases, the default is TCPIP. However,

for applications which use Language Environment services, the lack of a TCPIPJOBNAME statement causes applications that issue __iptcpn() to receive a *jobname* of NULL, rather than the default of TCPIP. Although this presents no problem when running in a single-stack environment, this can potentially cause errors in a multi-stack environment. The maximum length of the start procedure is 8 characters.

Examples

To specify TCPIPA as the name of the procedure that was used to start the TCP/IP address space, use the following code:

```
TCPIPJOBNAME TCPIPA
```

Usage notes

You must specify the proper procedure name of the TCP/IP address space on your system. If *tcpip_proc* is not the name of the started TCP/IP address space, applications using any TCP/IP provided API fail with an irrecoverable interaddress communication error.

For more information about why the TCPIPJOBNAME parameter must match the name of the associated TCP/IP address space and be the same name as that defined for the corresponding AF_INET physical file system in the BPXPRMxx member used to configure z/OS UNIX, see z/OS Communications Server: IP Configuration Guide.

TCPIPUSERID statement

The TCPIPUSERID statement is functionally equivalent to the TCPIPJOBNAME statement. See “TCPIPJOBNAME statement” on page 395.

TRACE RESOLVER statement

Use the TRACE RESOLVER statement to have a complete trace of all queries to and responses from the name server issued. Specifying TRACE RESOLVER is equivalent to the OPTIONS DEBUG statement.

Restrictions:

- The TRACE RESOLVER statement should not be specified in the GLOBALTCPIPDATA file. Before making a TCPIP.DATA file global, it should be tested for syntax errors. Trace output should appear in the SYSTCPT data set or RESOLVER_TRACE file that was specified.
- TRACE RESOLVER should be the first statement in the TCPIP.DATA statements to get the maximum trace output.

Syntax

```
►►─┬──────────────┬─TRACE RESOLVER──────────────────────────────────►◄
   └─system_name:─┘
```

Parameters

system_name:
The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

Steps for modifying

You can refresh this statement using the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Examples

To do a complete trace of all queries to and from the name server, use the following code:

```
TRACE RESOLVER
```

Usage notes

The TRACE RESOLVER statement is used for debugging purposes only.

Related topics

See z/OS Communications Server: IP Diagnosis Guide for information about interpreting and directing the output.

TRACE SOCKET statement

Use the TRACE SOCKET statement to have a complete trace of all calls to TCP/IP through the C socket library.

Syntax

```
►►─┬──────────────┬─TRACE SOCKET──────────────────────────────────►◄
   └─system_name:─┘
```

Parameters

system_name:
The name of the system to which this statement applies. See "system_name considerations" on page 370 for a complete description of this parameter.

Requirement: The colon is required.

Examples

Do a complete trace of all TCP/IP C socket calls:

```
TRACE SOCKET
```

Usage notes

The TRACE SOCKET statement is used for debugging purposes only.

The output from the TRACE SOCKET command is sent to the data set referred to by the SYSPRINT DD statement.

; and # statements

Use the ; or # to indicate a comment. Any data after the ; or # character is treated as a comment.

Sample TCPIP.DATA data set (TCPDATA)

The following shows sample TCPIP.DATA statements that can be used to configure information used by the resolver and TCP/IP application programs. The sample is shipped as member TCPDATA in the z/OS Communications Server SEZAINST data set.

```
;***********************************************************************
;                                                                      *
;   Name of Data Set:     TCPIP.DATA                                   *
;                                                                      *
;   COPYRIGHT = NONE.                                                  *
;                                                                      *
;   This data, TCPIP.DATA, is used to specify configuration            *
;   information required by TCP/IP client and server programs.         *
;                                                                      *
;                                                                      *
;   Syntax Rules for the TCPIP.DATA configuration data set:            *
;                                                                      *
;   (a) All characters to the right of and including a ; or # will     *
;       be treated as a comment.                                       *
;                                                                      *
;   (b) Blanks and <end-of-line> are used to delimit tokens.           *
;                                                                      *
;   (c) The format for each configuration statement is:                *
;                                                                      *
;       <SystemName||':'>  keyword  value                              *
;                                                                      *
;       where <SystemName||':'> is an optional label that can be       *
;       specified before a keyword; if present, then the keyword-      *
;       value pair will only be recognized if the SystemName matches   *
;       the name of the MVS system.                                    *
;       SystemName is derived from the MVS image name. Its value should*
;       be the IEASYSxx parmlib member's SYSNAME= parameter value.     *
;       The SystemName can be specified by either restartable VMCF     *
;       or the subsystem definition of VMCF in the IEFSSNxx member of  *
;       PARMLIB.                                                       *
;                                                                      *
;       For SMTP usage use the NJENODENAME statement in the SMTP       *
;       configuration data set to specify the JES nodename for mail    *
;       delivery on the NJE network.                                   *
;                                                                      *
;   (d) There should be no sequence numbers in this dataset. If there  *
;        are they can be treated as invalid statement parameters.      *
;                                                                      *
;***********************************************************************
;
; TRACE RESOLVER statement
; ========================
; TRACE RESOLVER will cause a complete trace of all queries to and
; responses from the name server or site tables.
; This command is for debugging purposes only.
; It should be the first statement in the TCPIP.DATA statements to get
; the maximum trace output.
;
; TRACE RESOLVER
;
;
; OPTIONS statement
; =================
; Use the OPTIONS statement to specify the following:
;  DEBUG
;   Causes resolver debug messages to be issued. This is equivalent to
;    TRACE RESOLVER. If used it should be the first statement in the
;    TCPIP.DATA statements to get the maximum trace output.
;  NDOTS:n
;   Indicates the number of periods (.) that need to be contained in a
;    domain name for it to be considered a fully qualified domain name
;
; OPTIONS NDOTS:1 DEBUG
;
;
; TCPIPJOBNAME statement
; ======================
; TCPIPJOBNAME specifies the name of the started procedure that was
; used to start the TCPIP address space.    TCPIP is the default for
; most cases.  However, for applications which use Language Environment
; services, the lack of a TCPIPJOBNAME statement causes applications
; that issue __iptcpn() to receive a jobname of NULL, and some of these
; applications will use INET instead of TCPIP.  Although this presents
; no problem when running in a single-stack environment, this can
; potentially cause errors in a multi-stack environment.
;
; If multiple TCPIP stacks are run on a single system, each stack will
; require its own copy of this file, each with a different value for
; TCPIPJOBNAME.
;
TCPIPJOBNAME TCPIP
;
;
; HOSTNAME statement
; ==================
; HOSTNAME specifies the TCP host name of this system as it is known
; in the IP network.  If not specified, the default HOSTNAME will be
; the name specified by either restartable VMCF or the subsystem
; definition of VMCF in the IEFSSNxx member of PARMLIB.
; If the VMCF name is not available then the IEASYSxx parmlib member's
; SYSNAME= parameter value will be used.
;
; For example, if this TCPIP.DATA data set is shared between 2
; systems, OURMVSNAME and YOURMVSNAME, then the following 2 lines
; will define the HOSTNAME correctly on each system.
;
; OURMVSNAME:    HOSTNAME  OURTCPNAME
; YOURMVSNAME:   HOSTNAME  YOURTCPNAME
;
; No prefix is required if the TCPIP.DATA file is not being shared.
;
; HOSTNAME THISTCPNAME
;
;
; NOTE - Use either DOMAINORIGIN/DOMAIN or SEARCH to specify your domain
;        origin value
;
; DOMAINORIGIN or DOMAIN statement
; ================================
; DOMAINORIGIN or DOMAIN specifies the domain origin that will be
; appended to host names passed to the resolver.  If a host name
; ends with a dot, then the domain origin will not be appended to the
; host name.
;
; DOMAINORIGIN  YOUR.DOMAIN.NAME
;
;
; SEARCH statement
; ================
; SEARCH specifies a list of 1 to 6 domain origin values that will be
; appended to host names passed to the resolver.  If a host name
; ends with a dot, then none of the domain origin values will be
; appended to the host name.
;  The first domain origin value specified by SEARCH will be used as the
; DOMAINORIGIN/DOMAIN value.
;
; SEARCH YOUR.DOMAIN.NAME my.domain.name domain.name
;
;
; DATASETPREFIX statement
; =======================
; DATASETPREFIX is used to set the high level qualifier for dynamic
; allocation of data sets in TCP/IP.
;
; The character string specified as a parameter on
; DATASETPREFIX takes precedence over the default prefix of "TCPIP".
;
; The DATASETPREFIX parameter can be up to 26 characters long
; and the parameter must NOT end with a period.
```

```
;
; For more information please see "Dynamic Data Set Allocation" in
; the IP Configuration Guide.
;
DATASETPREFIX TCPIP
;
;
; MESSAGECASE statement
; =====================
; MESSAGECASE MIXED indicates to some servers, such as FTPD, that
; messages should be displayed in mixed case.  MESSAGECASE UPPER
; indicates that all messages should be displayed in uppercase.  Mixed
; case strings that are inserted in messages will not be uppercased.
;
; If MESSAGECASE is not specified, mixed case messages will be used.
;
; MESSAGECASE MIXED
; MESSAGECASE UPPER
;
;
; NOCACHE statement
; =================
; NOCACHE specifies that resolver cache processing should not be used.
; If NOCACHE is not specified, then the current system-wide level of
; resolver caching is used.
;
; NOCACHE
;
;
; NSINTERADDR or NAMESERVER statement
; ===================================
; NSINTERADDR or NAMESERVER specifies the IP address of a name server.
;
; If you do not use name servers, then do not code any NSINTERADDR or
; NAMESERVER statements.  If you do have name servers, then code the
; IPv4 or IPv6 address of the remote name servers to be contacted.
;
; The NSINTERADDR or NAMESERVER statement can be repeated up to sixteen
; times to specify alternate name servers.  The name server listed first
; will be the first one attempted.
;
; IPv4 name server address:
; NSINTERADDR 10.0.0.1
;
; Ipv6 name server address:
; NSINTERADDR fc00::1
;
; NSPORTADDR statement
; ====================
; NSPORTADDR specifies the foreign port of the name server.
; 53 is the default value.
;
; NSPORTADDR 53
;
;
; RESOLVEVIA statement
; ====================
;
; RESOLVEVIA specifies how the resolver is to communicate with the
; name server.  TCP indicates use of TCP connections.  UDP indicates
; use of UDP datagrams.  The default is UDP.
;
RESOLVEVIA UDP
;
;
; RESOLVERTIMEOUT statement
; =========================
; RESOLVERTIMEOUT specifies the time that the resolver will wait for
; a response from the name server when using RESOLVEVIA UDP.
; The default is 5 seconds.
;
RESOLVERTIMEOUT 5
;
;
; RESOLVERUDPRETRIES statement
; ============================
;
; RESOLVERUDPRETRIES specifies the number of times the resolver
; should try to connect to the name server when using UDP datagrams.
; The default is 1.
```

```
;
RESOLVERUDPRETRIES 1
;
;
; LOOKUP statement
; ================
; LOOKUP indicates the order of name and address resolution.  DNS means
; use the DNSs listed on the NSINTERADDR and NAMESERVER statements.
; LOCAL means use the local host tables as appropriate for the
; environment being used (UNIX System Services or Native MVS).
;
; LOOKUP DNS LOCAL
;
;
; LOADDBCSTABLES statement
; ========================
; LOADDBCSTABLES indicates to the FTP server and FTP client which DBCS
; translation tables should be loaded at initialization time. Remove
; from the list any tables that are not required. If LOADDBCSTABLES is
; not specified, no DBCS tables will be loaded.
;
; LOADDBCSTABLES JIS78KJ JIS83KJ SJISKANJI EUCKANJI HANGEUL KSC5601
; LOADDBCSTABLES TCHINESE BIG5 SCHINESE
;
;
; SOCKDEBUG statement
; ===================
; Use the SOCKDEBUG statement to turn on the tracing of TCP/IP C and
; REXX socket library calls.
; This command is for debugging purposes only.
;
; SOCKDEBUG
;
;
; SOCKNOTESTSTOR statement
; ========================
; SOCKTESTSTOR is used to check socket calls for storage access errors
; on the parameters to the call.  SOCKNOTESTSTOR stops this checking
; and is better for response time.  SOCKNOTESTSTOR is the default.
;
; SOCKTESTSTOR
; SOCKNOTESTSTOR
;
;
; SORTLIST statement
; ==================
; Use the SORTLIST statement to specify the ordered list (maximum of 4)
; of network numbers (subnets or networks) for the resolver to prefer
; if it receives multiple addresses as the result of a name query.
;
; SORTLIST 128.32.42.0/24 128.32.42.0/255.255.0.0 9.0.0.0
;
;
; TRACE SOCKET statement
; ======================
; TRACE SOCKET will cause a complete trace of all calls to TCP/IP
; through the C socket library.
; This statement is for debugging purposes only.
;
; TRACE SOCKET
;
;
; ALWAYSWTO statement
; ===================
; ALWAYSWTO causes messages for some servers, such as SMTP and LPD,
; to be issued as WTOs. Specifying YES can cause excessive operator
; console messages to be issued.
;
ALWAYSWTO NO
; ALWAYSWTO YES
;
; Obsolete statements
; ===================
; The following statements no longer have any effect when included in
; this file:
;   SOCKBULKMODE
;   SOCKDEBUGBULKPERF0
;
; End of file.
;
```

Figure 7. Sample TCPIP.DATA data set (TCPDATA)

Chapter 6. z/OS Load Balancing Advisor and Load Balancing Agent

This topic contains the following information:

The z/OS Load Balancing Advisor communicates with outboard load balancers (LBs) and one or more z/OS Load Balancing Agents.

The purpose of the z/OS Load Balancing Advisor is to provide information to an outboard load balancer (such as a CISCO Content Switching Module [CSM]) about the availability of various resources (applications) and their relative ability to handle additional workload with respect to other resources that have the ability to handle the same workload. The outboard load balancer takes data the z/OS Load Balancing Advisor passes to it and makes a determination about where to route new workloads. This load balancing solution is different than existing load balancing solutions such as sysplex distributor and CISCO Multi-node Load Balancing (MNLB) in that in this implementation, the actual decision of where to route work is made outside of the sysplex.

For additional overview and configuration information about the z/OS Load Balancing Advisor, see z/OS Communications Server: IP Configuration Guide.

General syntax rules for z/OS Load Balancing Advisor

The following list shows the general configuration rules for the z/OS Load Balancing Advisor:

- Each statement must have a corresponding value and be separated from its value by one or more blanks.
- Only one statement and its values can be specified per line.
- Text beyond the specified statement and values is ignored.
- Text beginning with the # is a comment and is ignored. The remainder of the line following the # is considered part of the comment.
- Statements should be specified only once. When a statement is repeated, a warning message is written to the syslogd file, and the last instance of the statement is used.
- Statements that contain braces ({ and }) must specify the braces on separate lines. For example:

```
agent_id_list
{
1.2.3.4..8000
10.10.10.0..9000
}
```

Starting the z/OS Load Balancing Advisor

You must start the Advisor from a start procedure. A sample start procedure is shipped in member EZBLBADV in SEZAINST. The Advisor must have a configuration file. A sample Advisor configuration file is shipped in member EZBLBADC in SEZAINST.

Load Balancing Advisor sample start procedure

This topic shows the advisor sample start procedure.

```
//LBADV      PROC
//*
//*   IBM Communications Server for z/OS
//*   SMP/E distribution name: EZBLBADV
//*
//*   Licensed Materials - Property of IBM
//*   5694-A01
//*   Copyright IBM Corp. 2005, 2009
//*   Status = CSV1R11
//*
//*   Function: Sample procedure for running the
//*             z/OS Load Balancing Advisor
//*
//LBADV  EXEC PGM=EZBLBADV,REGION=0K,TIME=NOLIMIT,
//   PARM='/'
//*
//*** Notes:
//*
//* - The system link list concatenation must contain the TCP/IP
//*   runtime libraries and the C runtime libraries.  If they are
//*   not in the link list concatenation, this procedure will need
//*   to be changed to STEPLIB to them.
//*   If you add them to STEPLIB, they must be APF authorized.
//*
//* - The z/OS Load Balancing Advisor requires a configuration file
//*   which can be a member of an MVS PDS(E), an MVS sequential file,
//*   or a z/OS UNIX file.
//*
//CONFIG   DD DSN=TCPIP.TCPPARMS(LBADVCNF),DISP=SHR
//*CONFIG  DD DSN=TCPIP.CONFIG.LBADV,DISP=SHR
//*CONFIG  DD PATH='/etc/lbadv.conf',PATHOPTS=(ORDONLY)
//STDENV   DD DUMMY
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//CEESNAP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSMDUMP DD DISP=SHR,DSN=your.data.set.name
```

Figure 8. Advisor sample start procedure

Load Balancing Advisor configuration file statements

Table 10 on page 405 lists the advisor configuration file statements.

Rule: You must specify at least one of the load balancer connection statements (lb_connection_v4 or lb_connection_v6). You can specify both statements.

Table 10. Advisor configuration file statements

Configuration file statement	Default	Required or Optional	Purpose
agent_connection_port	No value is specified	Required	Specifies the port the Advisor should listen on for connections from Agents.
agent_id_list	No value is specified	Optional	Specifies which agents are allowed to connect to the Advisor.
debug_level	7	Optional	Specifies the level of debug information that is logged.
lb_connection_v4	No value is specified	Required (1)	Specifies the IPv4 address and port the Advisor should listen on for IPv4 connections from load balancers.
lb_connection_v6	No value is specified	Required (1)	Specifies the IPv6 address and port the Advisor should listen on for IPv6 connections from load balancers.
lb_id_list	No value is specified	Optional	Specifies which load balancers are allowed to connect to the Advisor.
port_list	No value is specified	Optional	Specifies a list of ports and the type of WLM recommendation that should be used for each.
sysplex_group_name	No value is specified	Optional	Specifies the TCP/IP sysplex group name for the subplex that this Advisor handles.
update_interval	`60`	Optional	Specifies how often agents update the Advisor with new information.
wlm	`basewlm`	Optional	Specifies the default type of WLM recommendation that should be used.

The connected arrows in Figure 9 on page 406 show configuration relationships relative to the Advisor. The IP address in the advisor_id statement can be any IP address belonging to the TCP/IP stack the Advisor is running on; however, this value should be a DVIPA.

Tip: The Agent host_connection statement and the corresponding entry in the Advisor agent_id_list are optional if AT-TLS is used for the connection between the Advisor and the Agent.

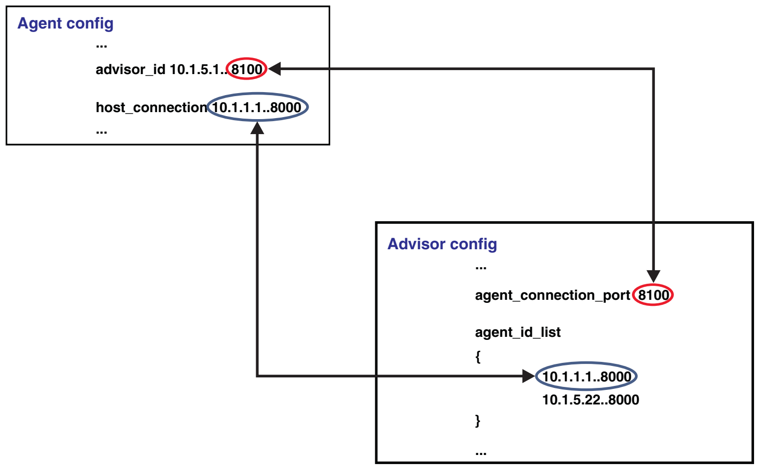


Figure 9. Advisor relationships

agent_connection_port statement

Use the agent_connection_port to specify the port the Advisor should listen on for connections from Agents.

Syntax

```
►►—agent_connection_port—host_port——————————————————————————————►◄
```

Parameters

host_port

Use *host_port* to specify which port the Advisor listens on for connections from Agents.

Requirement: This port must match the port number specified in the advisor_id configuration statement on Agents that wish to connect to this Advisor.
The valid range of port values is 1 - 65 535.

Restriction: If the host where the Advisor is running has only a single interface IP address, do not specify the same port on the agent_connection_port and lb_connection_v4 or lb_connection_v6 statements. Having only a single IP address means both the z/OS Load Balancer and the Agent would need to be configured with that address, and both would end up connecting to the Advisor's socket for z/OS Load Balancer connections.

agent_id_list statement

Use the agent_id_list statement to specify which agents are allowed to connect to the Advisor. This statement is optional.

Rules:

- If you are using AT-TLS with SERVAUTH access control checks to validate all Agent connections to this Advisor, this statement is optional. If you specify this statement, but AT-TLS is used for the connection, this statement is ignored. If you omit this statement, AT-TLS is required and the security checks must succeed.

- If you are not using AT-TLS with SERVAUTH access control checks to validate all Agent connections to this Advisor, this statement is required.

Syntax

```
                     ┌──────────────────────────────┐
                     ▼                              │
►►──agent_id_list──{────agent_ipaddr ..agent_port───┴──}──────────────────►◄
```

Parameters

agent_ipaddr..agent_port

Use *agent_ipaddr..agent_port* to specify a list of Agents that are allowed to connect to the Advisor. This parameter is required. The list consists of one or more blank-delimited IP address and port pairs each specified on a separate line, which are all contained within braces. There should be no spaces between the IP address, the two ellipses (..), and the port. These pairs represent the IP address and port of a given Agent.

There is no limit to the length of a line. The IP address can be an IPv4 or an IPv6 address. The IPv4 INADDR_ANY address (0.0.0.0) and the IPv6 unspecified address (::) are not allowed. If *agent_ipaddr..agent_port* is specified, this parameter must match the addresses specified in the host_connection configuration statement on the Agents.

The valid range of port values is 1 - 65 535.

Rule: You can specify only complete IP addresses. You cannot specify host names, prefixes, or subnets. Any given *agent_ipaddr..agent_port* pair must be specified on one line; it cannot be continued to a subsequent line.

debug_level statement

Use the debug_level statement to specify the level of debug information that is logged.

Syntax

```
►►──debug_level ──n──────────────────────────────────────────────────────►◄
```

Parameters

n Use *n* to specify the debug level. All log messages are written to syslogd. The value of *n* represents a particular debug level or combination of debug levels according to the following values:

0	None. No debug messages are logged.
1	Error-level messages are logged.
2	Warning-level messages are logged.
4	Event-level messages are logged.
8	Info-level messages are logged.

16 Message-level messages are logged. These are details of the messages (packets) sent between the Advisor and Load Balancer, and the Advisor. This level is intended for IBM service use only.

64 Debug-level messages are logged. These are internal debug messages intended for Development and Service. This level is intended for IBM service use only.

128 Trace-level messages are logged. These are function entry and exit traces that show the path through the code. This level is intended for IBM service use only.

Guideline: To log a combination of debug levels, add the debug level numbers. The default debug level is 7, which captures all Error, Warning, and Event messages.

lb_connection_v4 statement

Use the lb_connection_v4 statement to specify the IPv4 address and port the Advisor should listen on for IPv4 connections from load balancers.

This statement is optional. However, if neither an lb_connection_v4 nor an lb_connection_v6 statement is present in the configuration file, a terminating error results.

Syntax

```
                                    ┌─..3860──────┐
►►──lb_connection_v4 ──host_ipaddr──┼─────────────┼──────────────────────────────►◄
                                    └─..host_port─┘
```

Parameters

host_ipaddr..host_port

Use *host_ipaddr..host_port* to specify which IPv4 address and optionally the port the Advisor listens on for IPv4 connections from a load balancer. This address and port must be coordinated on any load balancers that wish to connect to this Advisor. The port is optional and defaults to 3860.

Rule: There should be no spaces between the IP address, the two ellipses (..), and the port. The *host_ipaddr..host_port* pair must be specified on one line. It cannot be continued to a subsequent line.

The valid range of port values is 1 - 65 535.

Guideline: This address should be a DVIPA.

Restriction: If the host where the Advisor is running has only a single interface IP address, do not specify the same port on the agent_connection_port and lb_connection_v4 or lb_connection_v6 statements. Having only a single IP address means both the z/OS Load Balancer and the Agent would need to be configured with that address, and both would end up connecting to the Advisor's socket for z/OS Load Balancer connections.

lb_connection_v6 statement

Use the lb_connection_v6 statement to specify the IPv6 address and port the Advisor should listen on for IPv6 connections from load balancers.

This statement is optional. However, if neither an lb_connection_v4 nor an lb_connection_v6 statement is present in the configuration file, a terminating error results.

Syntax

```
                                  ┌─..3860──────┐
►►──lb_connection_v6 ──host_ipaddr─┼─────────────┼──────────────────────────►◄
                                  └─..host_port─┘
```

Parameters

host_ipaddr..host_port

Use *host_ipaddr..host_port* to specify which IPv6 address and optionally the port the Advisor listens on for IPv6 connections from a load balancer. This address and port must be coordinated on any load balancers that wish to connect to this Advisor. The port is optional and defaults to 3860.

Rule: There should be no spaces between the IP address, the two ellipses (..) and the port. The *host_ipaddr..host_port* pair must be specified on one line. It cannot be continued to a subsequent line.

The valid range of port values is 1 - 65 535.

Guideline: For higher availability, specify a unique application-instance DVIPA.

Restrictions:

- If the host where the Advisor is running has only a single interface IP address, do not specify the same port on the agent_connection_port and lb_connection_v4 or lb_connection_v6 statements. Having only a single IP address means both the z/OS Load Balancer and the Agent would need to be configured with that address, and both would end up connecting to the Advisor's socket for z/OS Load Balancer connections.
- In general, most IPv6 listening sockets accept IPv4 connections if the listening socket is using the IPv6 unspecified address (::). However, for this listening socket, only IPv6 connections are accepted, even if you use the IPv6 unspecified address. If you expect to receive IPv4 connections from load balancers, you must specify the lb_connection_v4 statement.

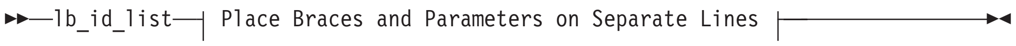
lb_id_list statement

Use the lb_id_list statement to specify which load balancers are allowed to connect to the Advisor.

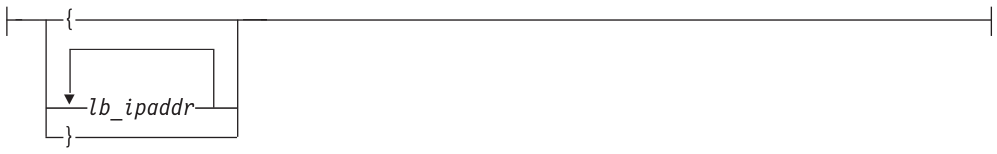
Rules:

- If you are using AT-TLS with SERVAUTH access control checks to validate all connections between Advisor-load balancer and Advisor-ADNR, this statement is optional. If you specify this statement, but AT-TLS is used for the connection, this statement is ignored. If this statement is not specified, AT-TLS is required and the security checks must succeed.
- If you are not using AT-TLS with SERVAUTH access control checks to validate all Advisor-load balancer connections and Advisor-ADNR connections, this statement is required or the connection is refused.

Syntax



Place Braces and Parameters on Separate Lines:



Parameters

lb_ipaddr
: Specifies the load balancers that are allowed to connect to the Advisor. The list consists of one or more blank-delimited IP addresses, each specified on a separate line. All IP addresses are contained in one set of braces. These addresses represent the IP address of a given load balancer. There is no limit to the length of a line.

Rules:

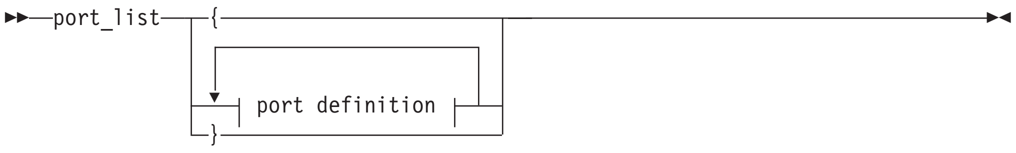
- You can specify only complete IP addresses. You cannot specify host names, prefixes, or subnets. Any IP address parameter on this statement must be specified on one line; it cannot be continued to a subsequent line.
- If you specify at least one IPv4 address, you must specify an lb_connection_v4 statement. Similarly, if you specify at least one IPv6 address, you must specify an lb_connection_v6 statement.

Guideline: If the load balancer has multiple source IP addresses that it can use, ensure that the lb_id_list statement contains the address that the load balancer should use as a source IP address when connecting as a client to the Advisor.

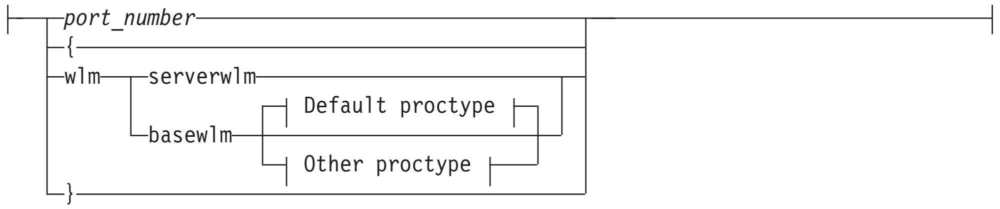
port_list statement

Use the port_list statement to specify a list of ports and the type of Workload Manager (WLM) recommendation that should be used for each.

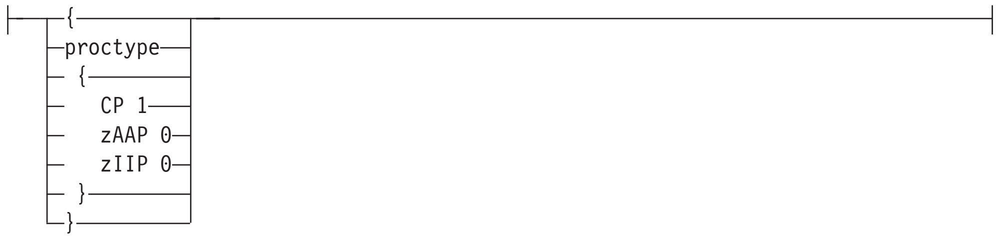
Syntax



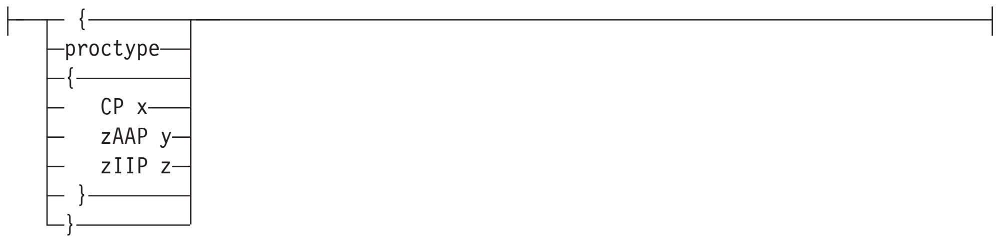
port definition:



Default proctype:



Other proctype:



Guideline: Place brackets and parameters on separate lines.

Parameters

port_number
A numerical value that represents a valid port number. Keywords on the remainder of this statement are applied to all members that match this port number.

wlm
A keyword that is used to override the default WLM recommendation method that is specified or made the default by the wlm statement. See z/OS Communications Server: IP Configuration Guide for more information about WLM recommendation types.

basewlm
Specifies that WLM system weight recommendations are being used for determining the best candidates for workload balancing for all members with ports that match the *port_number* value.

proctype

For workloads that use server-specific WLM weights, WLM typically returns a composite raw weight that takes into consideration how well the server is meeting its WLM goals with respect to the various types of processors the server is using.

For workloads that use WLM system weight recommendations, WLM is unaware of how a resource is using the various processors. Instead, WLM returns a weight for each processor type that is based on the amount of displaceable capacity for this processor in this system as compared to the available capacity for this processor on the other target systems.

For applications that use specialty processors and receive WLM system weight recommendations, specify the *proctype* parameter to indicate the expected proportion of each type of processor that the target application's workloads should use. A composite recommendation is determined from these proportions. Express each of the proportions as a number in the range 0 - 99. Each proportion value is divided by the total to determine the processor usage pattern; see the example that follows. To determine the processor proportions to configure, study your workload usage of assist processors by analyzing SMF records, using performance monitors reports, such as RMF, and so on.

For example, the *proctype* value CP 5 zAAP 0 zIIP 3 specifies a processor usage pattern such that 5/8 of the application's processor uses conventional processors (CP), and 3/8 of the application's CPU utilization uses zIIP processors.

For example, PROCTYPE CP 60 ZAAP 30 ZIIP 10, specifies a CPU usage pattern such that 60% uses conventional processors (CP), 30% uses zAAP processors, and 10% uses zIIP processors.

zAAPs and zIIPs are specialty processors designed to offload specific application workloads. Some target applications can take advantage of these specialty processors. For workloads that use server-specific WLM weights, WLM typically returns a composite raw weight that takes into consideration how well the server is meeting its WLM goals with respect to the various types of processors the server is using. For workloads that use the system-wide WLM recommendation, WLM is unaware of how a resource is using the various processors. Instead, WLM returns a weight for each processor type that is based on the amount of displaceable capacity for this processor in the system as compared to the available capacity for this processor on the other target systems.

Possible values are:

CP *x*
The proportion of the workload that uses conventional processors.

zAAP *y*
The proportion of the workload that uses zAAP processors.

zIIP *z*
The proportion of the workload that uses zIIP processors.

Requirement: If you specify a *proctype* value, it must be followed by braces (each brace on a separate line and each processor type and its value on a separate line). Each processor type parameter is optional; however, at least one processor type and its value must be coded. Each processor type can be specified in any order. When specified, each processor type must be specified with a value. Each processor type parameter and its value must be specified on separate lines.

Restrictions: When processor types are specified, at least one type must be specified with a nonzero value.

When you specify a *proctype* value on this statement, all *proctype* values on WLM statements are overridden. If you do not specify a processor type, the value 0 is assumed for that processor type.

serverwlm
Specifies that server-specific WLM recommendations are the WLM recommendation method to be used for determining the best candidates for workload balancing for all members with ports matching the *port_number* value. The actual WLM recommendation method used can be different than the configured method, depending on whether if each system reporting on members of the group supports server-specific WLM recommendations.

sysplex_group_name statement

Use the sysplex_group_name statement to specify the TCP/IP sysplex group name for the subplex that this Advisor handles when operating the Load Balancing Advisor in a sysplex subplexing environment.

You should specify a sysplex_group_name statement in the configuration file of each Load Balancing Advisor that is operating in a sysplex subplexing environment. The statement is optional. If the statement is omitted, it is assumed that the Advisor is not running in a subplexing environment.

Syntax

```
►►──sysplex_group_name──EZBTvvtt──────────────────────────────────────►◄
```

Parameters

EZBT*vvtt*
Specify the TCP/IP sysplex group name that is associated with the subplex that this Advisor handles. The TCP/IP sysplex group name is in the format EZBT*vvtt*. The *vv* value is the VTAM subplex group ID, as specified on the VTAM XCFGRPID start option. The *tt* value is the TCP/IP subplex group ID, as specified on the XCFGRPID parameter of the GLOBALCONFIG statement in the TCP/IP profile. If you did not specify the VTAM XCFGRPID start option when VTAM was started, then *vv* is CP. If you did not specify the XCFGRPID parameter on the GLOBALCONFIG statement in the TCP/IP profile, then *tt* is CS.

Tip: Use the DISPLAY TCPIP,,SYSPLEX,GROUP command to display the current TCP/IP sysplex group name.

update_interval statement

Use the update_interval statement to specify how often agents update the Advisor with new information.

Syntax

```
►►──update_interval ──n──────────────────────────────────────────────────►◄
```

Parameters

n Use *n* to specify the update interval in seconds, which determines how frequently Agents update the Advisor with information. *n* must be an integer in the range of 10 - 600 (10 seconds to 10 minutes). This statement is optional. The default is 60 seconds.

wlm statement

Use the wlm statement to specify the default Workload Manager (WLM) recommendation method that is used for each member, unless overridden on a per-port basis by the port_list statement. For more details about WLM recommendation types, see Customizing optional statements in z/OS Communications Server: IP Configuration Guide.

This statement is optional, and the default is basewlm.

Syntax

```
                        ┌─{ proctype { CP 1 zAAP 0 zIIP 0 } }─┐
           ┌─basewlm────┤                                     ├──┐
           │            └─{ proctype { CP x zAAP y zIIP z} }──┘  │
►►──wlm────┼─────────────────────────────────────────────────────┼────────────►◄
           └─serverwlm───────────────────────────────────────────┘
```

Parameters

basewlm
Specifies that WLM system weight recommendations should be used to determine the best candidates for workload balancing.

proctype
For workloads that use server-specific WLM weights, WLM typically returns a composite raw weight that takes into consideration how well the server is meeting its WLM goals with respect to the various types of processors the server is using.

For workloads that use WLM system weight recommendations, WLM is unaware of how a resource is using the various processors. Instead, WLM returns a weight for each processor type that is based on the amount of displaceable capacity for this processor in this system as compared to the available capacity for this processor on the other target systems.

For applications that use specialty processors and receive WLM system weight recommendations, specify the *proctype* parameter to indicate the expected proportion of each type of processor that the target application's workloads should use. A composite recommendation is determined from these proportions. Express each of the proportions as a number in the range 0 - 99. Each proportion value is divided by the total to determine the processor usage pattern; see the example that follows. To determine the processor proportions to configure, study your workload usage of assist processors by analyzing SMF records, using performance monitors reports, such as RMF, and so on.

For example, the *proctype* value CP 5 zAAP 0 zIIP 3 specifies a processor usage pattern such that 5/8 of the application's processor uses conventional processors (CP), and 3/8 of the application's processor uses zIIP processors.

zAAPs and zIIPs are specialty processors designed to offload specific application workloads. Some target applications can take advantage of these specialty processors. For workloads that use server-specific WLM weights, WLM typically returns a composite raw weight that takes into consideration how well the server is meeting its WLM goals with respect to the various types of processors the server is using. For workloads that use the system-wide WLM recommendation, WLM is unaware of how a resource is using the various processors. Instead, WLM returns a weight for each processor type that is based on the amount of displaceable capacity for this processor in the system as compared to the available capacity for this processor on the other target systems.

Possible values are:

CP *x* The proportion of the workload that uses conventional processors.

zAAP *y*
The proportion of the workload that uses zAAP processors.

zIIP *z* The proportion of the workload that uses zIIP processors.

Requirement: If you specify a *proctype* value, it must be followed by braces (each brace on a separate line and each processor type and its value on a separate line). Each processor type parameter is optional; however, at least one processor type and its value must be coded. Each processor type can be specified in any order. When specified, each processor type must be specified with a value. Each processor type parameter and its value must be specified on separate lines.

Restrictions: When processor types are specified, at least one type must be specified with a nonzero value.

When you specify a *proctype* value on this statement, all *proctype* values on WLM statements are overridden. If you do not specify a processor type, the value 0 is assumed for that processor type.

serverwlm
Specifies that server-specific WLM recommendations should be used to determine the best candidates for workload balancing. The actual WLM recommendation method used can be different than the configured method, depending whether each system that reports on members of the group supports server-specific WLM recommendations.

Result: zAAP and zIIP processor capacity is automatically included when the SERVERWLM parameter is specified and all systems in the sysplex are V1R9 or later.

Starting the z/OS Load Balancing Agent

Rule: You must start the Agent from a start procedure.

A sample start procedure is shipped in member EZBLBAGE in SEZAINST. A configuration file is required. A sample Agent configuration file is shipped in member EZBLBAGC in SEZAINST.

z/OS Load Balancing Agent sample start procedure

```
//LBAGENT    PROC
//*
//*   IBM Communications Server for z/OS
//*   SMP/E distribution name: EZBLBAGE
//*
//*   Licensed Materials - Property of IBM
//*   5694-A01
//*   Copyright IBM Corp. 2005, 2009
//*   Status = CSV1R11
//*
//*   Function: Sample procedure for running the
//*             z/OS Load Balancing Agent
//*
//LBAGENT EXEC PGM=EZBLBAGE,REGION=0K,TIME=NOLIMIT,
//   PARM='/'
//*
//*** Notes:
//*
//* - The system link list concatenation must contain the TCP/IP
//*   runtime libraries and the C runtime libraries.  If they are
//*   not in the link list concatenation, this procedure will need
//*   to be changed to STEPLIB to them.
//*   If you add them to STEPLIB, they must be APF authorized.
//*
//* - The z/OS Load Balancing Agent requires a configuration file
//*   which can be a member of an MVS PDS(E), an MVS sequential file,
//*   or a z/OS UNIX file.
//*
//CONFIG   DD DSN=TCPIP.TCPPARMS(LBAGECNF),DISP=SHR
//*CONFIG  DD DSN=TCPIP.CONFIG.LBAGENT,DISP=SHR
//*CONFIG  DD PATH='/etc/lbagent.conf',PATHOPTS=(ORDONLY)
//STDENV   DD DUMMY
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//CEESNAP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSMDUMP DD DISP=SHR,DSN=your.data.set.name
```

Figure 10. Agent sample start procedure

z/OS Load Balancing Agent configuration file statements

Table 11 lists the agent configuration file statements.

Table 11. Agent configuration file statements

Configuration file statement	Default	Required or Optional	Purpose
advisor_id	No value is specified	Required	Specifies the IP address and port of the Advisor that this Agent communicates with.

Table 11. Agent configuration file statements (continued)

Configuration file statement	Default	Required or Optional	Purpose
debug_level	7	Optional	Specifies the level of debug information that is logged.
host_connection	No value is specified	Optional	Specifies the local IP address and port that the Agent binds to for communications with the Advisor.
sysplex_group_name	No value is specified	Optional	Specifies the TCP/IP sysplex group name for the subplex that this Agent handles.

advisor_id statement

Use the advisor_id statement to specify the IP address and port of the Advisor that this Agent communicates with.

Syntax

```
►►──advisor_id ──advisor_ipaddr..advisor_port──────────────────────────────►◄
```

Parameters

advisor_ipaddr..advisor_port

Use *advisor_ipaddr..advisor_port* to specify the IP address and the port of the Advisor that this agent communicates with. Both the IP address and the port are required. The port must match the port specified in the Advisor's agent_connection_port configuration statement. The two ellipses (..) must immediately follow the *advisor_ipaddr* without any intervening spaces, and the port number must immediately follow the ellipses, without any intervening spaces. The IP address can be an IPv4 or an IPv6 address. The valid range of port values is 1 - 65 535.

Guideline: This address should be a VIPA.

Rule: If you specify an IPv4 address for advisor_id and you specify host_connection, you must specify an IPv4 address in the host_connection statement. Similarly, if you specify an IPv6 address for advisor_id and you specify host_connection, you must specify an IPv6 address in the host_connection statement.

debug_level statement

Use the debug_level statement to specify the level of debug information that is logged.

Syntax

```
►►—debug_level—n——————————————————————————————————————————————————————►◄
```

Parameters

n Use *n* to specify the debug level. All log messages are written to syslogd. The value of *n* represents a particular debug level or combination of debug levels according to the following values:

0	None. No debug messages are logged.
1	Error-level messages are logged.
2	Warning-level messages are logged.
4	Event-level messages are logged.
8	Info-level messages are logged.
16	Message-level messages are logged. These are details of the messages (packets) sent between the Advisor and Load Balancer, and the Advisor. This level is intended for IBM service use only.
32	Collection-level messages are logged. These are details of the collection and manipulation of data supporting the calculated weights. This level is intended for IBM service use only.
64	Debug-level messages are logged. This level is intended for IBM service use only.
128	Trace-level messages are logged. These are function entry and exit traces that show the path through the code. This level is intended for IBM service use only.

Guideline: To log a combination of debug levels, add the debug level numbers. The default debug level is 7, which captures all Error, Warning, and Event messages.

host_connection statement

Use the host_connection statement to specify the local IP address and port that the Agent binds to for communications with the Advisor.

Rules:

- If you are using AT-TLS with SERVAUTH access control checks to validate the Advisor-Agent connection, this statement is optional. If you specify this statement, but AT-TLS with SERVAUTH access control checks is used for the connection, the Advisor does not verify that the *host_ipaddr* and *host_port* specified on this statement is in its agent_id_list statement. If you omit this statement, AT-TLS with SERVAUTH access control checks is required, and the security checks must succeed.
- If you are not using AT-TLS with SERVAUTH access control checks to validate the Advisor-Agent connection, this statement is required, and the address and port must match one of the IP address and port pairs that are specified in the Advisor's agent_id_list configuration statement.
- If you specify this statement, the Agent binds to the specified IP address and port.

Syntax

```
►►──host_connection ──host_ipaddr..host_port─────────────────────────────────►◄
```

Parameters

host_ipaddr..host_port
Specifies the local IP address and the port that this Agent binds to. Both the IP address and the port are required. The two ellipses (..) must immediately follow the *host_ipaddr* value without any intervening spaces, and the port number must immediately follow the ellipses (..) without any intervening spaces. The IP address can be an IPv4 or an IPv6 address.

The valid range of port values is 1 - 65 535.

Rules:

- If you specify an IPv4 address, you must specify an IPv4 address in the advisor_id statement. Similarly, if you specify an IPv6 address, you must specify an IPv6 address in the advisor_id statement.
- In a subplexing environment, the address specified on the host_connection statement must be configured and owned by a stack in the same subplex as the Load Balancing Agent. If there is more than one TCP/IP stack on this system, in this subplex, the IP address must be a DVIPA defined in a VIPARANGE statement on each of these TCP/IP stacks so that the Agent can connect without regard to the order in which the TCP/IP stacks on this system are started.

sysplex_group_name statement

Use the sysplex_group_name statement to specify the TCP/IP sysplex group name for the subplex that this Agent handles when operating the Load Balancing Agent in a sysplex subplexing environment.

You should specify the sysplex_group_name statement in the configuration file of each Load Balancing Agent operating in a sysplex subplexing environment. The statement is optional. If the statement is omitted, it is assumed that the Agent is not running in a subplexing environment

Tip: Use the DISPLAY TCPIP,,SYSPLEX,GROUP command to display the current TCP/IP sysplex group name.

Syntax

```
►►──sysplex_group_name──EZBTvvtt──────────────────────────────────────────────►◄
```

Parameters

EZBT*vvtt*
Specify the TCP/IP sysplex group name that is associated with the subplex that this Agent handles. The TCP/IP sysplex group name is in the format EZBT*vvtt*. The *vv* value is the VTAM subplex group ID, as specified on the VTAM XCFGRPID start option. The *tt* value is the TCP/IP subplex group ID, as specified on the XCFGRPID parameter of the GLOBALCONFIG statement in the TCP/IP profile. If you did not specify the VTAM XCFGRPID start option

when VTAM was started, then *vv* is CP. If you did not specify the XCFGRPID parameter on the GLOBALCONFIG statement, then *tt* is CS.

Chapter 7. Automated domain name registration

The automated domain name registration (ADNR) application is a function that dynamically updates name servers with information about sysplex resources in near real time. The DNS names managed by ADNR can be names that represent all instances of an application within the sysplex, names that represent a specific instance of an application within the sysplex, names that represent the entire sysplex, as well as names that represent individual systems within the sysplex.

This topic contains the following information:
- "Starting the automated domain name registration application" on page 422
- "General configuration rules for automated domain name registration"
- "EZBADNRS sample start procedure for automated domain name registration application" on page 422
- "Automated domain name registration application configuration file" on page 424

For additional overview and configuration information about automated domain name registration, see z/OS Communications Server: IP Configuration Guide.

General configuration rules for automated domain name registration

The following list shows the general configuration rules for the automated domain name registration application (ADNR):
- Each statement must have a corresponding value and be separated from its value by one or more blanks.
- Only one statement and its values can be specified per line.
- Text beyond the specified statement and values is ignored.
- Statements that contain braces ({ and }) must specify the braces on separate lines. For example:

```
DNS mydns1
  {
   dns_id 10.11.12.0..553
     zone zone1
      {
       domain_suffix myplex1.mycorp.com
       transfer_key transfer_key1
       update_key update_key1
      }
     zone zone2
      {
       domain_suffix myplex2.mycorp.com
       transfer_key transfer_key2
       update_key update_key2
       ttl 15
      }
  }
```

- Any text beyond an opening or closing brace is ignored.
- Text beginning with a # is a comment and is ignored. The remainder of the line following the # is considered part of the comment.
- A uuid statement, gwm statement, and at least one dns statement are required.

- For statements with identical labels, a warning message is written to the log, and the last instance of the statement is used.
- As a statement is processed, all of the parameters are examined. Any parameter that is specified incorrectly causes an error. Any inconsistencies between parameters also cause an error. For example, using an IPv4 address for the local endpoint and an IPv6 address as a remote endpoint causes an error.
- Inconsistencies between statements cause errors. For example, some statements reference other statements. Statement order is not important, but if after processing the entire file if all references are not resolved (such as a dns statement referencing a key statement that is not present), an error results.
- When generating resource records (RRs), more than one group statement can refer to the same dns statement and zone parameter. However, each of the resource records generated for a zone must be a unique combination of *member_name* and *ipaddress* values. The name created when registering with a DNS server must be less than 255 characters. See the complete rules for name creation in the zone parameter in "dns statement" on page 428.
- A maximum of 100 zones is supported. If the file contains more than 100 zone statements, this causes an error.
- Files specified in any statements are opened and read to verify the existence of the file, and to verify that the correct permissions are in place to properly access the file.

Starting the automated domain name registration application

Rule: You must start the automated domain name registration (ADNR) application from a start procedure.

A sample start procedure is shipped in member ADNRS in SEZAINST. The ADNR application must have a configuration file. A sample configuration file is shipped in member ADNRC in SEZAINST.

EZBADNRS sample start procedure for automated domain name registration application

This topic shows a sample EZBADNRS start procedure.

```
//ADNR     PROC
//*
//*   IBM Communications Server for z/OS
//*   SMP/E distribution name: EZBADNRS
//*
//*   Licensed Materials - Property of IBM
//*   5694-A01
//*   Copyright IBM Corp. 2006,2009
//*   Status = CSV1R11
//*
//*   Function: Sample procedure for running the
//*             Automated Domain Name Registration application
//*
//ADNR   EXEC PGM=EZBADNR,REGION=0K,TIME=NOLIMIT,
//   PARM='/'
//*
//* To start ADNR with stack affinity:
//*ADNR   EXEC PGM=EZBADNR,REGION=0K,TIME=NOLIMIT,
//*   PARM=('ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPCS4")/')
//*
//*** Notes:
//*
//* - The system link list concatenation must contain the TCP/IP
//*   runtime libraries and the C runtime libraries.  If they are
//*   not in the link list concatenation, this procedure will need
//*   to be changed to STEPLIB to them.
//*   If you add them to STEPLIB, they must be APF authorized.
//*
//* - The Automated Domain Name Registration requires a configuration
//*   file which can be a member of an MVS PDS(E), an MVS sequential
//*   dataset, or a z/OS UNIX file.
//*
//CONFIG   DD DSN=TCPIP.TCPPARMS(ADNRCNF),DISP=SHR
//*CONFIG  DD DSN=TCPIP.CONFIG.ADNR,DISP=SHR
//*CONFIG  DD PATH='/etc/adnr.conf',PATHOPTS=(ORDONLY)
//STDENV   DD DUMMY
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//CEESNAP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSMDUMP DD DISP=SHR,DSN=your.data.set.name
```

Figure 11. EZBADNR start procedure

Automated domain name registration application configuration file

Table 12 lists the automated domain name registration application configuration file statements.

Table 12. Automated domain name registration application configuration (ADNR) file statements

Configuration file statement	Default	Default Required or Optional	Purpose	See
arm_element_suffix	none	Optional	Specifies the suffix added to the default element name for the automatic restart manager (ARM).	“arm_element_suffix statement” on page 426
debug_level	7	Optional	Specifies the level of debug information that is logged.	“debug_level statement” on page 426
dns	none	Required (1)	Specifies a DNS server that supports dynamic DNS registration.	“dns statement” on page 428
gwm	none	Required (2)	Specifies the Global Workload Manager (GWM) that advises the ADNR application.	“gwm statement” on page 430
host_group	none	Optional	Specifies the set of IP addresses to register for a group of hosts.	“host_group statement” on page 431
ipaddrlist	none	Optional (3)	Specifies a list of IP addresses being referenced from the host_group or server_group statements.	“ipaddrlist statement” on page 433
key	none	Optional	Specifies the key to use when signing a dynamic DNS update or zone transfer.	“key statement” on page 434
server_group	none	Optional	Specifies the set of IP addresses to register for a group of servers.	“server_group statement” on page 434

Table 12. Automated domain name registration application configuration (ADNR) file statements (continued)

Configuration file statement	Default	Default Required or Optional	Purpose	See
uuid	none	Required	Identifies this automated domain name registration application instance and distinguishes it from all other automated domain name registration application instances, as well as from all other external load balancers using SASP.	"uuid statement" on page 436
Notes: 1. At least one dns statement is required. 2. Only the last gwm statement in the configuration file is used. 3. Required if either a host_group or server_group statement is specified.				

Rule: If the configuration file has been changed any time after ADNR has been initially started, you must issue a MODIFY <*proc_name*> REFRESH command before ADNR is stopped.

The connected arrows in Figure 12 on page 426 show configuration relationships relative to the z/OS Load Balancing Advisor (LBA) Advisor application. The IP address in the *gwm_id* parameter can be any IP address belonging to the TCP/IP stack that the Advisor is running on; however, this value should be a dynamic VIPA (DVIPA). The IP address in the host_connection_addr parameter can be any IP address belonging to the TCP/IP stack that the automated domain name registration application is running on; this value should also be a DVIPA.

Tip: ADNR's *host_connection_addr* parameter and the corresponding entry in the Advisor's *lb_id_list* are optional if AT-TLS is used for the connection between the Advisor and ADNR.

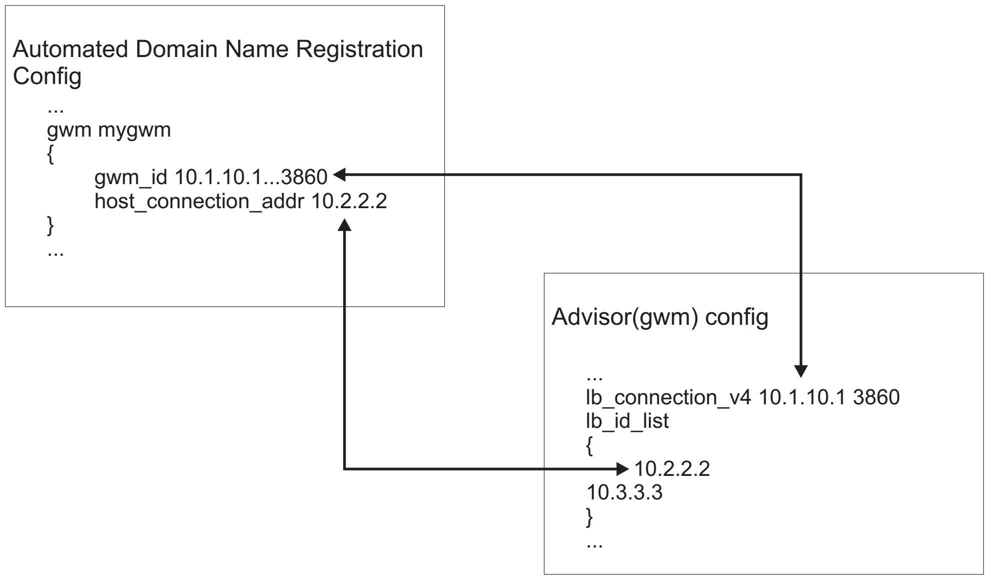

Automated Domain Name Registration Config
...
gwm mygwm
{
gwm_id 10.1.10.1...3860
host_connection_addr 10.2.2.2
}
...
Advisor(gwm) config
...
lb_connection_v4 10.1.10.1 3860
lb_id_list
{
10.2.2.2
10.3.3.3
}
...

Figure 12. Configuration Relationships

arm_element_suffix statement

Use the arm_element_suffix statement to specify the suffix added to EZBADNR for the automatic restart manager (ARM) element name. This statement is optional.

Syntax

```
►►──arm_element_suffix──name──────────────────────────────────────────►◄
```

Parameters

name

The arm_element_suffix value is an EBCDIC string 1 - 8 characters in length. The following characters are valid:

- Uppercase alpha characters
- Numeric characters (0 - 9)
- $, #, @, and underscore (_)

Rules:

- The name should be unique across the sysplex.
- Do not enclose the string in quotation marks.

Restriction: The number symbol (#) is not allowed as the first character of name.

debug_level statement

Use the debug_level statement to specify the type and quantity of logging information to be written to the log file.

Syntax

```
     ┌─debug_level 7─┐
►►───┤               ├──────────────────────────────────────────────►◄
     └─debug_level n─┘
```

Parameters

n Used to specify the debug level. All log messages are written to syslogd. The value of *n* represents a particular debug level or combination of debug levels according to the values shown in Table 13.

To log a combination of debug levels, add together the debug level numbers. If a debug level value is not specified, the default debug level is 7, which captures all ERROR, WARNING, and EVENT messages.

Table 13 lists the possible values:

Table 13. debug_level values

Debug level	Logging level	Description
0	None	No messages of any kind are sent to the logging file after initialization is complete.
1	ERROR	Error messages indicate that something requires attention. Messages at this level can be fatal (terminating) or can indicate that an important part of the automated domain name registration application is not working properly. This information is logged at the syslogd ERROR priority level.
2	WARNING	Warning messages indicate that an error has occurred, but it is not severe enough to warrant an ERROR level. Corrective action might be necessary because the automated domain name registration application might not be functioning as intended. This information is logged at the syslogd WARNING priority level.
4	EVENT	Event messages are logged for events that occur periodically, like operator commands, UNIX signals, timer pops, receipt of a network message, and so on. This information is logged at the syslogd NOTICE priority level.
8	INFO	Informational messages are sent to the logging file. These messages do not require corrective action. This information is logged at the syslogd INFO priority level.

Table 13. debug_level values (continued)

Debug level	Logging level	Description
16	MESSAGE	Messages about the detailed contents of message packets that are sent between the GWM (Advisor) and the automated domain name registration application. Use these messages to assist debugging automated domain name registration application communications. This information is logged at the syslogd DEBUG priority level.
32	COLLECTION	Collection messages concern the details of managing the data in the DNS zones. This information is logged at the syslogd DEBUG priority level.
64	DEBUG	Debug messages are intended for Development or Service and give detail that customers do not normally want. The intention of this level of message is to provide information that is useful in debugging code, logic, or timing errors. This information is logged at the syslogd DEBUG priority level.
128	TRACE	Trace messages are intended for Development or Service to track code processing (footprints). This information is logged at the syslogd DEBUG priority level.

dns statement

Use the dns statement to define a DNS server that supports dynamic DNS update. The automated domain name registration application update-adds and update-deletes host names and IP addresses with the DNS server.

Requirement: This statement is required.

Syntax

```
►►──dns ──dns_label──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
                                            ┌──────────┐
                         ┌─..53────────┐    ▼          │
├──{──dns_id ──dns_addr──┼─────────────┼──┤ zone ├──┴──}──────────────────────┤
                         └─..dns_port──┘
```

zone:

```
├──zone zone_label──{──domain_suffix──domain_suffix──┬────────────────────┬──────►
                                                      └─update_key key_label─┘
```

```
                                         ┌─ttl gwm_update_interval─┐
►──┬──────────────────────────────┬──────┼─────────────────────────┼──}──────────────────────┤
   └─transfer_key key_label───────┘      └─ttl ttl─────────────────┘
```

Parameters

dns
The keyword that defines the beginning of the dns statement.

dns_label
A string 1 - 32 characters in length for the label of this DNS server. This value is referenced from other statements in the automated domain name registration application configuration file and from automated domain name registration commands.

dns_id *dns_addr..dns_port*
Used to specify the IP address and port this DNS server is listening on. The port is optional; the default value is 53.

The valid range of port values is 1 - 65 535.

Rules:

- Do not put any spaces between the IP address, the two ellipses (..), and the port.
- The *dns_addr..dns_port* value must be specified on one line. It cannot be continued to a subsequent line.

zone
The keyword that defines the beginning of a zone owned by this DNS server.

zone_label
A string 1 - 32 characters in length for the label that specifies the name of the zone for which the DNS server is authoritative. This value is referenced from other statements in the automated domain name registration application configuration file and from automated domain name registration commands.

Restriction: Within the configuration file, all zone labels must be unique.

domain_suffix *domain_suffix*
The domain suffix of a zone for which the DNS server is authoritative.

Rules:

- All domain suffixes must be unique within a dns statement.
- The domain suffix cannot contain two consecutive periods together.

When the *domain_suffix* value is created, a trailing period is added if one was not configured.

All dynamic DNS updates sent to the DNS server have the domain suffix appended. Thus, the following names can be created by the automated domain name registration application. The longest of these names must be less than or equal to 255 characters in length, including the trailing period. For example:

```
host_group_name.domain_suffix.
host_name.domain_suffix.
server_group_name.domain_suffix.
server_name.server_group_name.domain_suffix.
```

update_key *key_label*
: The label referencing the key statement that represents the key used in signing update-add and update-delete requests sent to this DNS server. If the update_key parameter is not specified, meaning transaction signature (TSIG) is not being used, then the update-add and update-delete requests are not signed.

transfer_key *key_label*
: The label referencing the key statement that represents the key used in signing zone transfer requests sent to this DNS server. If the transfer_key parameter is not specified, meaning transaction signatures (TSIG) is not being used, then the zone transfer requests are not signed.

ttl
: The time to live (TTL) value used for domain names registered with this zone. This indicates the amount of time, in seconds, that a DNS record exists in the cache of a non-authoritative name server or resolver before expiring from the cache. If the ttl parameter is not specified, the TTL is the value specified on the Load Balancing Advisor (LBA) *update_interval* configuration parameter. For information about the update interval of the LBA, see "Load Balancing Advisor configuration file statements" on page 404.

  The time to live of the resource record (RR) field is a 32-bit integer value, in seconds, and is primarily used by resolvers and non-authoritative name servers when they cache RRs. The TTL specifies how long an RR can be cached before it should be discarded.

  The valid range of ttl values is 0 - 2 147 483 647 seconds.

gwm statement

Use the gwm statement to define a Global Workload Manager (GWM) that advises the automated domain name registration application.

The z/OS Load Balancing Advisor—Advisor application is an instance of a GWM.

Requirement: This statement is required.

Restriction: Only the last gwm statement in the automated domain name registration application configuration file is used.

Syntax

```
►►──gwm ──gwm_label──┤ Put Braces and Parameters on Separate Lines ├──────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
                       ┌..3860─────┐
├──{──gwm_id ──gwm_addr──┴..gwm_port─┴──┬─────────────────────────────────┬──}─────────┤
                                        └─host_connection_addr ip_addr─┘
```

Parameters

gwm
: The keyword that defines the beginning of the gwm statement.

gwm_label

A string 1 - 32 characters in length for the label of the GWM that is communicating with the automated domain name registration application. This value is not referenced from other statements in the automated domain name registration application configuration file or from automated domain name registration commands.

gwm_id *gwm_addr..gwm_port*

The remote IP address and port that the GWM is listening on for connections from load balancers. The port is optional; the default value is 3860.

The valid range of port values is 1 - 65 535.

Tip: For higher availability, specify a unique application instance DVIPA for the remote IP address.

Rule: There should be no spaces between the IP address, the two ellipses (..), and the port. The *gwm_addr..gwm_port* value must be specified on one line. It cannot be continued to a subsequent line.

host_connection_addr *ipaddr*

The local IP address that the automated domain name registration application uses when creating the socket that is used to connect to the GWM if AT-TLS is not used.

Rules:

- If you are using AT-TLS with SERVAUTH access control checks to validate the Advisor-ADNR connection, this statement is optional. If you specify the host_connection_addr statement and use AT-TLS with SERVAUTH access control checks for the connection, the Advisor does not verify that the host_connection_addr statement is in its lb_id_list statement. If you omit this statement, AT-TLS with SERVAUTH access control checks is required, and the security checks must succeed.
- If you are not using AT-TLS with SERVAUTH access control checks to validate the Advisor-ADNR connection, this parameter is required and the IP address must match one of the IP addresses specified in the Advisor's lb_id_list configuration statement.
- If this parameter is specified, ADNR binds to the specified IP address.

Tip: For increased availability, specify a unique application-instance DVIPA for the local IP address.

host_group statement

Use the host_group statement to identify the set of IP addresses to update for a group of hosts. The automated domain name registration application updates the name server with the intersection between the IP addresses configured to the automated domain name registration application and the IP addresses active on the hosts in the sysplex.

The DNS names dynamically added to the name server take the form *host_group_name.domain_suffix*, where the *host_group_name* value is the name of the group of hosts being registered to the GWM and the *domain_suffix* value is the domain suffix name specified in a zone parameter on a dns statement.

The automated domain name registration application can also update individual host instances with the DNS server using the member keyword. The automated domain name registration application updates the name server with the intersection between the IP addresses configured for the member and the set of IP addresses active on the hosts in the sysplex. The DNS names dynamically added to the name server take the form host_name.domain_suffix, where *host_name* is the name of the member being registered to the GWM and *domain_suffix* is the domain suffix name specified in a zone parameter on a dns statement.

See the description of *domain_suffix* in “dns statement” on page 428 for total length restrictions.

Syntax

```
►►──host_group ──host_group_label──┤ Put Braces and Parameters on Separate Lines ├────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────────────────────────────────────────────┬──────────┤
   │  ┌────────────────┐                                                         │
   │  ▼                │                                                         │
   ├────┤ member ├─────┴──host_group_name──host_group_name──dns──dns_label──zone──zone_label─┤
   └─}───────────────────────────────────────────────────────────────────────────┘
```

member:

```
                                               ┌──────────────────────────────────┐
                                               ▼                                  │
├──member──{──┬────────────────────────────┬──────ipaddrlist ──ipaddrlist_label──┴──}────┤
              └─host_name ──host_name──────┘
```

Parameters

host_group
The keyword that defines the beginning of the host_group statement.

host_group_label
A string 1 - 32 characters in length for the label of this host group. This value is referenced from automated domain name registration commands.

host_group_name *host_group_name*
The name of the group of hosts to be updated in the name server. This is the default host name for a member defined without a *host_name* parameter.

Restrictions:

- The name must be less than or equal to 63 characters in length.
- The name cannot contain any periods.
- Within the entire configuration file, a group name must be unique. The *host_group_name* value cannot be used on any other host_group statements or be used on any other server_group statements as a *server_group_name* value.

dns *dns_label*
A label referencing the dns statement that defines the DNS server with which to register the domain name and IP addresses. The value specified matches the *dns_label* on a dns statement.

zone *zone_label*
A label referencing a zone on the DNS server that is identified by the dns keyword. The value specified matches the *zone_label* value on a dns statement.

member
The members for a given group. The member might refer to a single TCP/IP host instance or might apply to the host group itself.

host_name *host_name*
The name of the individual host to be updated in the name server. If the *host_name* value is not specified, the member statement applies to the host group itself and not an individual host.

Rules:

- Only one member can be defined for a host group without a host_name definition.
- The name must be less than or equal to 63 characters in length.
- The name cannot contain any periods.

ipaddrlist *ipaddrlist_label*
A label referencing an ipaddrlist statement. This list contains one or more IP addresses to register. These IP addresses can be IPv4 or IPv6 addresses.

Guideline: For increased availability, specify VIPAs to identify the host.

ipaddrlist statement

Use the ipaddrlist statement to define a set of IP addresses that are referenced by members of host_group and server_group statements in the automated domain name registration application configuration file.

Syntax

```
►►──ipaddrlist ──ipaddrlist_label──┤ Put Braces and Parameters on Separate Lines ├────────►◄
```

Put Braces and Parameters on Separate Lines:

```
         ┌────────────────┐
         ▼                │
├──{──────ipaddr ipaddr───┴──}─────────────────────────────────────────────┤
```

Parameters

ipaddrlist
The keyword that defines the beginning of the ipaddrlist statement.

ipaddrlist_label
A string 1 - 32 characters in length for the label of this ipaddrlist statement. This value is referenced from other statements in the automated domain name registration application configuration file.

ipaddr *ipaddr*
A single IP address to register. Multiple *ipaddr* values can be specified if there is more than one IP address associated with a host group or server group.

Guideline: These IP addresses can be a combination of IPv4 and IPv6 addresses.

key statement

Use the key statement to define the key name and key file to use when creating signatures for specifying transaction signatures (TSIGs) for zone updates and zone transfers.

Syntax

```
►►──key ──key_label──┤ Put Braces and Parameters on Separate Lines ├──────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──{──keyfile file_name──}──────────────────────────────────────────────────┤
```

Parameters

key
The keyword that defines the beginning of the key statement.

key_label
A string 1 - 32 characters in length for the label of the key used in creating signatures. This name is referenced by the update_key or transfer_key keywords on a dns statement in the automated domain name registration application configuration file.

keyfile *file_name*
The file that contains the shared secret used in creating signatures.

Rules:

- The *file_name* value must be a fully qualified name of a z/OS UNIX file.
- Both the .key and the .private key files generated by the dnssec-keygen utility must be available for TSIG authentication to work correctly, even though only the .key key file name is specified by the *file_name* value.
- The file name is case sensitive. For TSIG authentication to work properly, the file name extensions must be .key and .private.

server_group statement

Use the server_group statement to identify the set of IP addresses to update for a group of servers. The automated domain name registration application updates the name server with the intersection between the IP addresses configured to the automated domain name registration application and the set of IP addresses on which the servers are listening.

The DNS names dynamically added to the name server take the following form `server_group_name.domain_suffix`, where *server_group_name* is the name of the individual server instance being registered to the GWM, *server_group_name* is the name of the group of servers, which includes *server_name*, and *domain_suffix* is the domain suffix name specified in a zone parameter on a dns statement.

The automated domain name registration application can also update individual server instances with the DNS server using the member keyword. The automated domain name registration application updates the name server with the intersection between the IP addresses configured for the member and the set of IP addresses on which the server is listening. The DNS names dynamically added to the name server take the form `server_name.server_group_name.domain_suffix`, where *server_name* is the name of the individual server instance being registered to the GWM, *server_group_name* is the name of the group of servers which includes *server_name*, and *domain_suffix* is the domain suffix name specified in a zone parameter on a dns statement.

See the description of *domain_suffix* in "dns statement" on page 428 for total length restrictions.

Syntax

```
►►──server_group──server_group_label──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
        ┌──────────┐
        ▼          │
├──{────┤ member ├─┴──port port──protocol protocol──────────────────────►

►──server_group_name server_group_name──dns dns_label──zone zone_label──}──────────┤
```

member:

```
                                                   ┌──────────────────────────────────┐
                                                   ▼                                  │
├──member──{──┬──────────────────────────────┬──────ipaddrlist ──ipaddrlist_label─────┴──────────┤
              └──server_name ──server_name───┘
```

Parameters

`server_group`
The keyword that defines the beginning of the server_group statement.

server_group_label
A string 1 - 32 characters in length for the label of this server group. This value is referenced from automated domain name registration commands.

`port` *`port`*
The port on which the server is listening.

The valid range of port values is 1 - 65535.

`protocol` *`protocol`*
The transport protocol used by the application. This value must be TCP or UDP.

`server_group_name`
The name of the group of servers to be registered with DNS server. This is the default server name for a member defined without a *server_name* parameter.

Rules:

- The name must be less than or equal to 63 characters in length.
- The name cannot contain any periods.
- Within the entire configuration file, a group name must be unique. The *server_group_name* value cannot be used on any other server_group statements or be used on any other host_group statements as a *host_group_name* value.

`dns` *`dns_label`*
A label referencing a dns statement that defines the DNS server with which to register the domain name and IP addresses. The value specified matches the *dns_label* value on a dns statement.

`zone` *`zone_label`*
A label referencing a zone on the DNS server that is identified by the dns keyword. The value specified matches a *zone_label* value on a dns statement.

`member`
The members for a given group. The member might refer to a single TCP/IP server instance or it might apply to the server group itself.

`server_name` *`server_name`*
The name of the individual server to be registered with the DNS server. If this parameter is not specified, then the member statement applies to the server group itself and not an individual server. Specifying a server_name value enables clients to connect to a particular instance of a server that is a member of a server group.

Rules:

- Only one member can be defined for a server group without a server_name definition.
- The name must be less than or equal to 63 characters in length.
- The name cannot contain any periods.

`ipaddrlist` *`ipaddrlist_label`*
A label referencing an ipaddrlist statement. This list contains one or more IP addresses to register for this server or server group.

Guidelines:

- These IP addresses can be IPv4 or IPv6 addresses.
- For increased availability, specify DVIPAs and VIPAs to identify the server or server group.

uuid statement

Use the uuid [(Universally Unique ID (UUID)] statement to uniquely identify this automated domain name registration application instance and distinguish it from all other SASP external load balancers. The GWM uses this unique user ID to distinguish one SASP entity from another.

Requirement: This statement is required.

Syntax

```
►►──uuid──uuid──────────────────────────────────────────────────────────►◄
```

Parameters

uuid
The keyword that defines the beginning of the uuid statement.

uuid The *uuid* value is an EBCDIC string 1 - 64 characters in length. Do not enclose the string in quotation marks.

Chapter 8. IKE daemon

This topic contains the following information:

Starting the IKED using z/OS UNIX

Start the IKED from the z/OS shell using the following syntax:

```
►►──iked───────────────────────────────────────────────────────────────────►◄
```

Tip: When you are starting the IKE daemon from the z/OS UNIX shell, set the environment variable _BPX_JOBNAME. This enables a specific job name to be used when reserving ports for the IKE daemon. You can also use this name with the STOP or MODIFY console commands. For more information about _BPX_JOBNAME, see z/OS UNIX System Services Planning.

IKE cataloged procedure

This topic shows the IKE cataloged procedure.

Update the cataloged procedure, IKED, by copying the sample in SEZAINST(IKED), to your system or recognized PROCLIB. Specify IKE daemon parameters and change the data set names to suit your local configuration. See SEZAINST(EZARACF) for SAF considerations for started procedures. After the IKED procedure has been started, a different IKED configuration file can be specified by using the Modify command with the FILE parameter. For example:

```
MODIFY IKED,REFRESH,FILE='/etc/security/iked.conf2'
```

```
//IKED     PROC
//*
//* IBM Communications Server for z/OS
//* SMP/E distribution name: EZBIKPRC
//*
//* 5650-ZOS Copyright IBM Corp. 2005, 2013
//* Licensed Materials - Property of IBM
//* "Restricted Materials of IBM"
//* Status = CSV2R1
//*
//*
//IKED     EXEC PGM=IKED,REGION=0K,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/'
//*
//* Provide environment variables to run with the desired
//* configuration.  As an example, the data set or file specified by
//* STDENV could contain:
//*
//*   IKED_FILE=/etc/security/iked.conf2
//*   IKED_CTRACE_MEMBER=CTIIKE01
//*   IKED_CODEPAGE=IBM-1047
//*
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//* For information on the above environment variables, refer to the
//* IP Configuration Reference.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.IKED.ENV(IKED),DISP=SHR
//* Sample HFS file containing environment variables:
//*STDENV   DD PATH='/etc/security/iked.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

Figure 13. IKE cataloged procedure

IKE environment variables

Table 14 provides a list of environment variables used by IKE that can be tailored to a particular installation.

Table 14. IKE environment variables

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules
IKED_CODEPAGE	Server	Used by the IKE daemon to specify the EBCDIC code page to be used for the configuration file. The default code page is IBM-1047.	The following code pages are supported: • IBM-037 • IBM-273 • IBM-274 • IBM-275 • IBM-277 • IBM-278 • IBM-280 • IBM-281 • IBM-282 • IBM-284 • IBM-285 • IBM-297 • IBM-500 • IBM-871 • IBM-1047 • IBM-1140 • IBM-1141 • IBM-1142 • IBM-1143 • IBM-1144 • IBM-1145 • IBM-1146 • IBM-1147 • IBM-1148 • IBM-1149 Example: `IKED_CODEPAGE=IBM-1141`
IKED_FILE	Server	Used by the IKE daemon in the search order for the IKE daemon configuration file. For details on the search order used for locating this configuration file, see Table 1 on page 2	Example: `IKED_FILE=/etc/security/iked.conf`

Table 14. IKE environment variables (continued)

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules
IKED_CTRACE_MEMBER	Server	Used by the IKE daemon to specify the name of a parmlib member that contains default CTRACE settings. The IKED_CTRACE_MEMBER environment variable is read by the IKE daemon only during initialization. Changes to the IKED_CTRACE_MEMBER after daemon initialization have no effect.	If not defined, the default value used by the IKE daemon is CTIIKE00. Example: `IKED_CTRACE_MEMBER=CTIIKE00`

IKE daemon configuration file statements

If you specify a configuration file for the IKE daemon, the file must contain an IkeConfig statement. If you are using network security services, this file might also contain one NssStackConfig statement for each z/OS network security services (NSS) client TCP/IP stack. If you specify multiple IkeConfig statements, the last one is used; if you define multiple NssStackConfig statements for the same stack name, the last one is used.

If no configuration file is specified, then defaults are provided for IkeConfig parameters where possible. However, because there are no reasonable defaults for the NssStackConfig statements or the z/OS network security services (NSS) server-related parameters of the IkeConfig statement, it is not possible to have any NSS client TCP/IP stacks. All stacks in this case are handled using local services available to the IKE daemon.

If a configuration error is detected during startup, then the IKE daemon logs the error and exits. If a configuration error is detected during a dynamic refresh, then the entire refresh is rejected, the error is logged, and the IKE daemon continues running with the old configuration values.

Tip: The terms phase 1 and phase 2 refer to different types of security associations (SAs) that the z/OS IKE daemon can negotiate with its peers. Although the specific terminology for these types of security associations differs between the IKE version 1 and IKE version 2 protocols, the terms phase 1 and phase 2 refers to both versions, as shown in Table 15.

Table 15. IKE terminology: phase 1 and phase 2

Term	Usage in regard to IKE protocol version
Phase 1 security association (SA)	Refers to IKE version 1 phase 1 SAs as well as IKE version 2 IKE SAs. When a specific version is intended, that version is identified in this document.
Phase 2 security association (SA)	Refers to IKE version 1 phase 2 SAs as well as IKE version 2 child SAs. When a specific version is intended, that version is identified in this document.

IkeConfig statement

If you code more than one IkeConfig statement, the last statement is used. Likewise, if a parameter other than SMF119 or SupportedCertAuth in the IkeConfig statement is specified more than once, the value from the last statement is used. SMF119 adds to, but does not replace, the types of SMF records to be written. SupportedCertAuth is used to define a set of certificate authorities (CAs) this value adds to, but does not replace, the list of CAs supported by a local security endpoint.

Syntax

```
►►──IkeConfig──┤ Braces &     Parms on Separate Lines ├──────────────────────►◄
```

Braces & Parms on Separate Lines:

```
├──┬─{────────────────────────────────────────────────────────────────────────┤
   ├─┤ IkeConfig Parameters ├─┤
   └─}────────────────────────┘
```

IkeConfig Parameters:

```
   ┌─IkeSyslogLevel──1─┐  ┌─PagentSyslogLevel──0─┐  ┌─SMF119──None──────┐  ┌─IkeRetries 6─┐
├──┼───────────────────┼──┼──────────────────────┼──┼───────────────────┼──┼──────────────┼──►
   └─IkeSyslogLevel──n─┘  └─PagentSyslogLevel──n─┘  ├─SMF119──None──────┤  └─IkeRetries n─┘
                                                    ├─SMF119──IKEALL────┤
                                                    ├─SMF119──IKETunnel─┤
                                                    └─SMF119──DynTunnel─┘

   ┌─IkeInitWait 2─┐  ┌─FIPS140 no──────┐  ┌─KeyRing──iked/keyring────┐  ┌─Echo──no──────┐
►──┼───────────────┼──┼─────────────────┼──┼──────────────────────────┼──┼───────────────┼──►
   └─IkeInitWait n─┘  └─FIPS140──┬─yes─┬┘  ├─KeyRing──userid/ringname─┤  └─Echo──┬─yes─┬┘
                                 └─no──┘   └─KeyRing──ringname────────┘          └─no──┘

►──┬──────────────────────────────────────────────────────────────────────────┬──────►
   │                              ┌─Port 4159─┐                               │
   └─NetworkSecurityServer host───┼───────────┼─Identity──┬─IpAddr authid─────┘
                                  └─Port port─┘           ├─Fqdn authid───────┤
                                                          ├─UserAtFqdn authid─┤
                                                          └─X500dn authid─────┘

►──┬──────────────────────────────────────────────────────────────────────────────┬──►
   │                                    ┌─Port 4159─┐                             │
   └─NetworkSecurityServerBackup host───┼───────────┼─Identity──┬─IpAddr authid───┘
                                        └─Port port─┘           ├─Fqdn authid───────┤
                                                                ├─UserAtFqdn authid─┤
                                                                └─X500dn authid─────┘

   ┌─NssWaitLimit 60───────┐  ┌─NssWaitRetries 3─┐  ┌─PagentWait──0─┐
►──┼───────────────────────┼──┼──────────────────┼──┼───────────────┼────────────────►
   └─NssWaitLimit seconds──┘  └─NssWaitRetries n─┘  └─PagentWait──n─┘

   ┌─────────────────────────┐
   ▼                         │
►──SupportedCertAuth Label───┴──}─────────────────────────────────────────────────┤
```

Parameters

IkeSyslogLevel

Specifies the level of logging to obtain from the IKE daemon. The following levels are supported:

0 - IKE_SYSLOG_LEVEL_NONE
Disable IKE daemon syslog messages

1 - IKE_SYSLOG_LEVEL_MINIMUM
Minimal IKE daemon syslog output

2 - IKE_SYSLOG_LEVEL_SADETAIL
Always output detailed Security Association (SA) information when available

4 - IKE_SYSLOG_LEVEL_DEBUGSA
Debug for SA negotiations

8 - IKE_SYSLOG_LEVEL_FMTPKTTRC
Formatted packet trace

16 - IKE_SYSLOG_LEVEL_UNFPKTTRC
Unformatted packet trace

32 - IKE_SYSLOG_LEVEL_VERBOSE
Show cascaded error messages

64 - IKE_SYSLOG_LEVEL_CERTINFO
Show certificates in CA cache when cache is initially built or rebuilt

128 Reserved

To specify a combination of log levels, add the level numbers. For example, to request FMTPKTTRC (8) messages and VERBOSE (32) messages, specify IkeSyslogLevel 40. Use the MODIFY IKED,REFRESH command to change this value. Level values greater than 1 are intended for diagnostic purposes only. A non-zero PagentSyslogLevel will take effect only if IkeSyslogLevel is also set to a non-zero value, otherwise no debug trace records are generated.

Rules:

- The default IkeSyslogLevel is in effect until the parameter is read from the configuration file.
- Any level higher than 1 automatically includes 1.

PagentSyslogLevel

Specifies the level of diagnostic logging to obtain for the interaction between the IKE daemon and the Policy Agent. The following levels are supported:

0 - PAGENT_SYSLOG_LEVEL_NONE
No logging of IKE daemon interactions with the Policy Agent.

1 - PAGENT_SYSLOG_LEVEL_EMERG
A panic condition

2 - PAGENT_SYSLOG_LEVEL_ALERT
Requires immediate action

4 - PAGENT_SYSLOG_LEVEL_CRIT
Critical condition

8 - PAGENT_SYSLOG_LEVEL_ERR
Error messages

16 - PAGENT_SYSLOG_LEVEL_WARNING
Warning messages

32 - PAGENT_SYSLOG_LEVEL_NOTICE
Conditions that are not error conditions, but might require special handling

64 - PAGENT_SYSLOG_LEVEL_INFO
Informational messages

128 - PAGENT_SYSLOG_LEVEL_DEBUG
Messages that contain information normally of use only when debugging a program

To specify a combination of log levels, add the level numbers. For example, to request LEVEL_EMERG (1) messages and LEVEL_WARNING (16) messages, specify PagentSyslogLevel 17. Use the MODIFY IKED,REFRESH command to change this value. Level values greater than 0 are intended for diagnostic purposes only. A non-zero PagentSyslogLevel will take effect only if IkeSyslogLevel is also set to a non-zero value, otherwise no debug trace records will be generated.

SMF119
Specifies the types of SMF 119 records to be written to the MVS SMF data sets. The following levels are supported:

None No SMF 119 records should be written to the MVS SMF data sets. This is the default.

IKEAll
All SMF 119 records should be written to the MVS SMF data sets. This setting includes all of the SMF 119 record types listed in this topic.

IKETunnel
SMF record type 119 subtypes related to phase 1 SA events should be written (subtypes 73 and 74) to the MVS SMF data sets.

DynTunnel
SMF record type 119 subtypes related to phase 2 SA events should be written (subtypes 75 and 76) to the MVS SMF data sets.

To specify a combination of records to be written, specify multiple SMF119 statements. Use the MODIFY IKED,REFRESH command to change this value.

KeyRing
The owning *userid* and *ringname* used by the IKE server when performing RSA signature mode of authentication. When using a key ring owned by IKE server, specify the ring name as *ringname*. When using a key ring owned by another user, specify the ring name as *userid/ringname*.

The KeyRing parameter is not used by NSS client TCP/IP stacks.

IkeRetries
Specifies the number of times that an unanswered IKE negotiation message is retransmitted before the negotiation is terminated. The value of *n* can be in the range 1 - 8. The default is six retransmissions (254 seconds before dropping the message exchange if the default IkeInitWait value of two seconds is used). The IKE server uses an exponentially increasing wait interval between each retransmission. The initial wait interval is specified by the IkeInitWait parameter, and each subsequent wait interval is doubled from there. For example, if the IkeInitWait value is two, the first retransmission comes after

two seconds, the second comes four seconds after the first, the fourth eight seconds after the third, and so on. Use the MODIFY IKED,REFRESH command to change this value.

Table 16 illustrates how a retransmission scenario would occur using the default values of IkeRetries 6 and IkeInitWait 2. The following scenario assumes that the IKE partner never responds to the IKE message in question.

Table 16. Example of an IkeRetries retransmission scenario

Event	**Seconds since last event**	**Elapsed time in seconds**
Send initial message	0	0
1st wait interval expires: message retransmitted	2	2
2nd wait interval expires: message retransmitted	4	6
3rd wait interval expires: message retransmitted	8	14
4th wait interval expires: message retransmitted	16	30
5th wait interval expires: message retransmitted	32	62
6th wait interval expires: message retransmitted	64	126
7th wait interval expires: message exchange is dropped	128	254 (See note)
Note: * 4 minutes, 14 seconds		

Table 17 illustrates how retransmission scenario would occur using the maximum values of IkeRetries 8 and IkeInitWait 15. This scenario assumes that the IKE partner never responds to the IKE message in question.

Table 17. Example of an IkeRetries retransmission using maximum values scenario

Event	**Seconds since last event**	**Elapsed time in seconds**
Send initial message	0	0
1st wait interval expires: message retransmitted	15	15
2nd wait interval expires: message retransmitted	30	45
3rd wait interval expires: message retransmitted	60	105
4th wait interval expires: message retransmitted	120	225
5th wait interval expires: message retransmitted	240	465
6th wait interval expires: message retransmitted	480	945
7th wait interval expires: message retransmitted	960	1905
8th wait interval expires: message retransmitted	1920	3825

Table 17. Example of an IkeRetries retransmission using maximum values scenario (continued)

Event	Seconds since last event	Elapsed time in seconds
9th wait interval expires: message exchange is dropped	3840	7665
Note: * 2 hours, 7 minutes, 45 seconds		

Table 18 illustrates how retransmission scenario would occur using the minimum values of IkeRetries 1 and IkeInitWait 1. This scenario assumes that the IKE partner never responds to the IKE message in question:

Table 18. Example of an IkeRetries retransmission using minimum values scenario

Event	Seconds since last event	Elapsed time in seconds
Send initial message	0	0
1st wait interval expires: message retransmitted	1	1
2nd wait interval expires: message exchange is dropped	2	3

IkeInitWait

Specifies the number of seconds to wait before the first retransmission of an unanswered IKE message. The value of *n* can be in the range 1 - 15. The default is 2 seconds. Use the MODIFY IKED,REFRESH command to change this value.

FIPS140

Specifies whether the IKE daemon should perform cryptographic operations by invoking cryptographic modules that are designed to meet the Level 1 security requirements documented in the Federal Information Processing Standard (FIPS) publication 140 (FIPS 140).

yes Perform all IKE daemon cryptographic operations using cryptographic modules that are designed to meet FIPS 140 requirements. When the value of yes is specified, the IKE daemon server is running in FIPS 140 mode.

no IKE daemon might perform some cryptographic operations using cryptographic modules that do not adhere to the FIPS 140 requirements. When the value of no is specified, the IKE daemon is not running in FIPS 140 mode.

Requirement: ICSF must be active before starting the IKE daemon when FIPS140 YES is specified. For information about configuring ICSF to support FIPS 140-2, see Operating in compliance with FIPS 140-2 in z/OS Cryptographic Services ICSF Writing PKCS #11 Applications.

Rule: This parameter cannot be modified while the IKED is running. Attempts to modify the value while the IKED is running are ignored and a warning message is issued.

Tip: Enabling FIPS 140 mode provides a higher degree of assurance of the integrity of the cryptographic modules that IKE uses, including ICSF and System SSL. However, enabling FIPS 140 mode might require additional setup and configuration, it will restrict the available set of cryptographic algorithms, and it might result in a reduction in performance. See Cryptographic standards and FIPS 140 in z/OS Communications Server: IP Configuration Guide for more information.

Echo

Echoes all IKE daemon log messages to the job output file, specified by the IKEDOUT DD (JCL) statement. Use the MODIFY IKED,REFRESH command to change this value.

NetworkSecurityServer

Identifies the primary NSS server for IKE NSS client TCP/IP stacks.

A single server is used for all of the TCP/IP stacks configured as NSS clients. Stacks can be configured individually as NSS clients. Stacks with a corresponding NssStackConfig statement are treated as NSS clients; stacks without a corresponding NssStackConfig statement rely solely on local IKE resources.

Tip: The NetworkSecurityServer parameter is optional. However, if both the NetworkSecurityServer and NetworkSecurityServerBackup parameters are not specified, none of the TCP/IP stacks can function as an NSS client.

Use the MODIFY IKED,REFRESH command to change this value. If you change the NetworkSecurityServer value, the changes take effect for new connections, but existing connections are not dropped. If you want the old connections to be dropped, perform the following steps:

1. Comment out the following statements:
   - NetworkSecurityServer statement (if present)
   - NetworkSecurityServerBackup statement (if present)
   - NssStackConfig statements (if present)
2. Issue a MODIFY IKED, REFRESH command to reread the IKED configuration file.
3. Uncomment the following statements:
   - NetworkSecurityServer statement (if present)
   - NetworkSecurityServerBackup statement (if present)
   - NssStackConfig statements (if present)
4. Issue a MODIFY IKED,REFRESH command to re-read the IKED configuration file.

host
The address of the NSS server can be specified either as a host name, a numeric IPv4 address, or a numeric IPv6 address. This is a required parameter. If a host name is specified, the maximum length accepted is 255 characters. The host name value should conform to the naming standards set forth by RFC 1035. For information about RFC, see Appendix C, "Related protocol specifications," on page 1465.

Examples of supported host identifiers are as follows:

```
163.44.212.11
1080:0:0:0:8:800:200C:417A
norton.nycsanitation.gov
```

Port *port*
The TCP port on which the NSS server is listening for connections from the IKE daemon. The default value is 4159. Valid values are in the range 1 - 65535.

This parameter is optional.

Identity

The identity of the NSS Server. This is a required parameter.

The IKE daemon requires that communication with an NSS Server be protected using AT-TLS. During the AT-TLS handshake, the NSS server provides a certificate that is used to authenticate its identity. The IKE daemon interrogates this certificate and verifies that the identity in the certificate matches the identity specified on the NetworkSecurityServer parameter of the IkeConfig statement.

The following identity types (for *idtype*) and formats (for *authid*) are supported:

IpAddr

Indicates that the *authid* value is a numeric IPv4 address or a numeric IPv6 address. For example, 1.2.3.4.

Fqdn

Indicates that the *authid* value is a fully qualified domain name or host name. For example, vnet.ibm.com. The maximum length accepted is 255 characters. The Fqdn value should conform to the naming standards set forth by RFC 1035.

UserAtFqdn

Indicates that the *authid* value is a user at a fully qualified domain name or host name. The user name cannot contain a blank. For example, ibm@vnet.ibm.com. The maximum length accepted is 512 characters. The UserAtFqdn value should conform to the naming standards set forth by RFC 822.

X500dn

Indicates that the *authid* value is an X.500 distinguished name (DN). The DN must be specified in accordance with RFC 2253. A double-byte character is represented using the escaped UTF-8 encoding of the double-byte character in the Unicode character set. Attribute types can be specified using either attribute names or numeric object identifiers. Attribute values must represent string values.

Any distinguished name that contains an imbedded blank must be enclosed in double quotes. For example, `X500dn "CN=R. Kramden,T=Driver,O=Gotham Bus Company,C=US"`.

Table 19 lists the DN attribute names that are recognized by the System SSL run time. An error is returned if the DN contains an unrecognized attribute name.

Table 19. DN attribute names

Abbreviation	Meaning
C	Country
CN	Common name
DC	Domain component
E	E-mail address
EMAIL	E-mail address (preferred)
EMAILADDRESS	E-mail address
L	Locality
O	Organization name
OU	Organizational unit name

Table 19. DN attribute names (continued)

Abbreviation	Meaning
PC	Postal code
S	State or province
SN	Surname
SP	State or province
ST	State or province (preferred)
STREET	Street
T	Title

The following code is an example of a DN using attribute names and string values:

```
CN=Hoffman,OU=Endicott,O=IBM,C=US
```

The following code is the same DN using object identifiers and encoded string values. The encoded string values represent the ASN.1 DER encoding of the string. The System SSL run time supports the following ASN.1 string types: PRINTABLE, VISIBLE, TELETEX, IA5, UTF8, BMP, and UCS.

```
2.5.4.3=#130E526F6E616C6420486F66666D616E,2.5.4.11=
#1308456E6469636F7474, 2.5.4.10=#130349424D,2.5.4.6=#13025553
```

Individual characters can be represented using escape sequences. This is useful when the character cannot be represented in a single-byte character set. The hexadecimal value for the escape sequence is the UTF-8 encoding of the character in the Unicode character set. Table 20 shows some Unicode example letter descriptions.

Table 20. Unicode letter descriptions

Unicode letter description	10646 code	UTF-8	Quoted
LATIN CAPITAL LETTER L	U0000004C	0x4C	L
LATIN SMALL LETTER U	U00000075	0x75	u
LATIN SMALL LETTER C WITH CARON	U0000010D	0xC48D	\C4\8D
LATIN SMALL LETTER I	U00000069	0x69	i
LATIN SMALL LETTER C WITH ACUTE	U00000107	0xC487	\C4\87

Guideline: The letters in the Quoted column in Table 20 can be used to encode a surname as follows:

```
SN=Lu\C4\8Di\C4\87
```

An escape sequence can also be used for special characters that are part of the name and are not to be interpreted as delimiters. The following special characters must be represented as an escape sequence (prefixed with a backslash [\]) when used as part of the name:

- A space or number sign (#) character occurring at the beginning of the string
- A space occurring at the end of the string
- One of the following characters , + " \ < >

This correct escape sequence is shown in the following example:

```
"CN=L. Eagle,OU=Jones\, Dale and Mian,O=IBM,C=US"
```

In this example, the enclosing double quotes are required because of the imbedded blanks, not because of the escaped characters.

Rule: When an X500dn type identity is specified, the DN attributes must have the same order as those of the corresponding certificate subject name.

NetworkSecurityServerBackup

Identifies the backup NSS server for the IKE daemon. The NSS server (or its backup) supplies certificate and remote management services for managed stacks.

A single backup server is used for all of the TCP/IP stacks configured as NSS clients.

The NetworkSecurityServerBackup parameter is optional. It allows network security clients to connect to a backup NSS server at a different address or port from the primary. Alternatively, in a sysplex configuration, the primary NSS server can be configured on a dynamic VIPA to use the recovery capabilities of dynamic addressing. If no backup server is available when the primary server is not responsive, certificate and remote management services are unavailable to network security clients. However, if a NetworkSecurityServerBackup parameter is not specified, then certificate services are unavailable to Network Security clients if the primary NSS server becomes unresponsive.

Network Security clients switch between the primary and the backup NSS servers whenever their current server becomes unresponsive. If both the primary and the backup become unresponsive, the Network Security client attempts to connect to the primary and the backup in a round-robin fashion until a successful connection is made. It is possible to have a situation where one NSS client is being managed by the primary server and another NSS client is being managed by the backup server. It is also possible to specify a backup server without specifying a primary server, in which case, the backup server is treated as if it is the primary server.

Use the MODIFY IKED,REFRESH command to change this value. If you change the NetworkSecurityServerBackup value, then the changes take effect for new connections, but existing connections are not dropped. If you want the old connections to be dropped, follow this following sequence:

1. Comment out all of the following statements:
   - NetworkSecurityServer statement (if present)
   - NetworkSecurityServerBackup statement (if present)
   - NssStackConfig statements (if present)
2. Issue a MODIFY IKED, REFRESH command to re-read the IKED configuration file.
3. Uncomment out all of the following statements:
   - NetworkSecurityServer statement (if present)

- NetworkSecurityServerBackup statement (if present)
- NssStackConfig statements (if present)

4. Issue a MODIFY IKED, REFRESH command to re-read the IKED configuration file.

host The address of the NSS server can be specified either as a host name, a numeric IPv4 address, or a numeric IPv6 address. This is a required parameter. If a host name is specified, the maximum length accepted is 255 characters. The host name value should conform to the naming standards set forth by RFC 1035.

Examples of supported host identifiers are as follows:

```
163.44.212.11
1080:0:0:0:8:800:200C:417A
norton.nycsanitation.gov
```

Port *port*
The TCP port on which the backup NSS server is listening for connections from the IKE daemon. The default value is 4159. Valid values are in the range 1 - 65535. This parameter is optional.

Identity
The identity of the backup NSS server. This is a required parameter.

The IKE daemon requires that communication with an NSS server be protected using AT-TLS. During the AT-TLS handshake the NSS server provides a certificate that is used to authenticate its identify. The IKE daemon interrogates this certificate and verifies that the identity in the certificate matches the identity specified on the NetworkSecurityServer parameter of the IkeConfig statement.

The following identity types (*idtype*) and formats (*authid*) are supported:

IpAddr
Indicates that the *authid* value is a numeric IPv4 address or a numeric IPv6 address. For example, 1.2.3.4.

Fqdn Indicates that the *authid* value is a fully qualified domain name or host name. For example, vnet.ibm.com. The maximum length accepted is 255 characters. The Fqdn value should conform to the naming standards set forth by RFC 1035.

UserAtFqdn
Indicates that the *authid* value is a user at a fully qualified domain name or host name. The user name cannot contain a blank. For example, ibm@vnet.ibm.com. The maximum length accepted is 512 characters. The UserAtFqdn value cannot begin or end with a dot (.) or contain consecutive dots. The UserAtFqdn value should conform to the naming standards set forth by RFC 822.

X500dn
Indicates that *authid* is an X.500 distinguished name (DN). See the NetworkSecurityServer parameter description in this topic for the DN specification.

NssWaitLimit
Specifies the number of seconds (1-300) that an NSS client waits between connection attempts when trying to establish a connection with an NSS server.

The product of the NssWaitLimit value multiplied by the NssWaitRetries value defines the maximum number of seconds that an NSS client attempts to connect to an NSS server before switching to another server. For example, if the NssWaitLimit value is 60, and the NssWaitRetries value is 3, then an NSS client waits at most for a total of 180 seconds for a successful connection with a given server. See the description of the NetworkSecurityServerBackup parameter for a discussion of how NSS clients switch between the primary and backup NSS servers.

The default value is 60 seconds. Use the MODIFY IKED,REFRESH command to change this value. The new value takes effect immediately.

NssWaitRetries

Specifies the number of times (1-10) that an NSS client attempts to establish a connection with an NSS server.

The product of the NssWaitLimit value multiplied by the NssWaitRetries value defines the maximum number of seconds that an NSS client attempts to connect to an NSS server before switching to another server. For example, if the NssWaitLimit value is 60, and the NssWaitRetries value is 3, then an NSS client waits at most for a total of 180 seconds for a successful connection with a given server. See the description of the NetworkSecurityServerBackup parameter for a discussion of how NSS clients switch between the primary and backup NSS servers.

The default value is 3 retries. Use the MODIFY IKED,REFRESH command to change this value. The new value takes effect immediately.

PagentWait

The time limit in seconds to wait for connection to the Policy Agent. The value of *n* can be 0-9999. A value of 0 indicates retry forever. The default is 0.

SupportedCertAuth

Specifies the label of a certificate on the IKE server's key ring. This label corresponds to the certificate of a certificate authority supported by the local security endpoint when using RSA signature mode of authentication. RSA signature authentication is a certificate-based authentication method used by the IKE server to authenticate a remote security endpoint's identify. The SupportedCertAuth parameter can be specified multiple times to identify a set of supported certificate authorities.

Use the SupportedCertAuth parameter to define a set of certificate authorities (CAs) supported by the local security endpoint. This list is provided to the remote security endpoint to request that it choose a certificate signed by an acceptable CA. The remote security endpoint is not constrained to choose certificates signed by CAs accepted by the local security endpoint. However, if the remote security endpoint chooses a certificate signed by a CA that is not on the IKE server's key ring, the key exchange fails.

The CaLabel parameter of the RemoteSecurityEndpoint IPSec policy statement can be used to further restrict the set of certificate authorities that can sign the certificate used by a particular remote security endpoint. The advantage of further restricting the set of certificate authorities that might sign the certificate used by a particular remote security endpoint is a reduction in the size of the IKE key exchange messages transmitted between the local security endpoint and the remote security endpoint.

The number of specified labels is limited to a maximum of 128. The maximum length of a label is 32 characters, which corresponds to the maximum length of a RACF label. The default is an empty list containing no labels.

Use the MODIFY IKED,REFRESH command to change this value.

The SupportedCertAuth parameter is not used by NSS server client TCP/IP stacks.

NssStackConfig statement

The NssStackConfig statement contains NSS server stack configuration information for the IKE daemon. Only stacks with a corresponding NssStackConfig statement are eligible for management services provided by network security services. Stacks that are not configured with an NssStackConfig statement do not use network security services.

Restriction: NssStackConfig statements require that a valid NSS server is set up in the IkeConfig statement. See the NetworkSecurityServer and NetworkSecurityServerBackup parameters in the IkeConfig statement. It is a configuration error to have an NSSStackConfig statement without also specifying a NetworkSecurityServer parameter, a NetworkSecurityServerBackup parameter, or both.

If more than one NssStackConfig statement is coded for the same TCP/IP stack, the last one is used. Likewise, if a parameter within the NssStackConfig statement is specified more than once, the value from the last one is used.

Use the MODIFY IKED,REFRESH command to change which TCP/IP stacks are configured as NSS clients, as follows:

Deleting an NSS client
If it is determined after a refresh that an NSSStackConfig statement was removed, then the connection associated with the removed NssStackConfig statement is closed

Adding NSS client
If it is determined after a refresh that a new NssStackConfig statement was added, then the connection for the new stack is opened.

Changing internal NssStackConfig values
Any change to an internal parameter of the NssStackConfig statement results in a disconnect followed by a reconnect.

Syntax

```
►►──NssStackConfig──stackname──┤ Braces &    Parms on Separate Lines ├──────────►◄
```

Braces & Parms on Separate Lines:

```
├──┬──────────────────────────────┬──────────────────────────────────────────┤
   ├─{────────────────────────────┤
   ├─ClientName ──clientname──────┤
   │  ┌────────────────────────┐  │
   │  ▼                        │  │
   ├───ServiceType──┬─RemoteMgmt─┬┴─┤
   │                └─Cert───────┘  │
   ├─UserId userid────────────────┤
   ├─AuthBy──┬─Password password─┬┤
   │         └─Passticket────────┘│
   └─}────────────────────────────┘
```

Parameters

`stackname`
The name of the NSS client TCP/IP stack. This is a required parameter. There is no default value.

`ClientName` `clientname`
The NSS client name for the stack. By default, client names have the form *sysname_stackname*, where the *sysname* value is the MVS system name, and the *stackname* value is the TCP/IP stack name. This name must match the clientname portion of the associated SERVAUTH profile (EZB.NSS.sysname.clientname.IPSEC.CERT and EZB.NSS.sysname.clientame.IPSEC.NETMGMT) and can be 1 - 24 characters in length.

Restriction: Only alphanumeric characters (a-z, A-Z, 0-9), the hyphen (-), and the underscore (_) are valid for the ClientName parameter. Embedded spaces are also not permitted in the ClientName parameter; only trailing spaces are permitted.

If no client name is configured, then the IKE daemon generates this parameter based on the system's host name and the associated TCP/IP stack name.

For example, if the system host name is MVSIBM and the TCP/IP stack name is TCPCS, then the generated client name is MVSIBM_TCPCS.

`ServiceType`
The ServiceType parameter should be specified once for each network security service that is to be enabled for the stack. The following service types are supported:

RemoteMgmt
Indicates that this stack is eligible for remote management.

Cert Indicates that this stack uses centralized certificate management.

Requirement: There must be as least one ServiceType statement in the NssStackConfig statement.

`UserId` `userid`
The RACF user ID that the NSS server uses to authenticate the NSS client and to verify its access to the SERVAUTH profiles that protect the certificate and remote management resources on the NSS server. User IDs can be 1 - 8 characters in length.

`AuthBy`
Authorization of the client TCP/IP stack to the NSS server can be accomplished either by the use of a password or by a Pass Ticket.

Password *password*
The *password* value is the RACF password for the user ID specified for the user. There is no default value for the *password* value; a valid password is required if password authentication is being used. Passwords can be 1 - 8 characters in length.

Passticket
The Pass Ticket option causes the client to generate a one-time session key. See the information about the secured signon function in z/OS Security Server RACF Security Administrator's Guide.

Authby is a required parameter and there is no default value. Either the Password option or Passticket option (but not both), must be specified.

During the installation, ensure that you prevent access to the IKE configuration file by unauthorized users to protect this sensitive data. The most secure approach to protecting this information is to use Pass Tickets, which store the application keys in the RACF database.

IKE daemon configuration file sample

```
#
# IBM Communications Server for z/OS
# SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZAIKCFG
#
# 5694-A01 Copyright IBM Corp. 2007 - 2010.
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# Status = CSV1R12
#
# /etc/security/iked.conf  (IKE daemon configuration)
#
# This file contains sample IKE daemon configuration parameters.
# The search order used by the IKE daemon to locate the initial
# configuration file is (highest priority listed first):
#
# 1) The name of a file or MVS data set specified by the IKED_FILE
#    environment variable.
# 2) /etc/security/iked.conf
#
# Some parameters may be dynamically modified after the
# IKE daemon has been started.  The parameters that are
# dynamically modifiable are noted below.
#
# One way of dynamically modifying parameters is to edit
# the iked.conf file after the IKE daemon has been started and then
# issue a modify command to cause the IKE daemon to re-read the file.
#
# Example: MODIFY IKED,REFRESH
# Note: IKED is the IKE daemon procedure name.
#
# After the IKE daemon has been started, a different configuration
# file can be specified by using the Modify command with the FILE
# parameter.  This allows modifiable parameters to be
# dynamically altered while the IKE daemon is running.  Note that
# the parameter values modified in this fashion are not
# persistent.  To make the changes persistent, edit the iked.conf
# file that is located at IKE initialization time according to the
# search order described previously.
#
# Example: MODIFY IKED,REFRESH,FILE='/etc/security/iked.conf2'
# Note: IKED is the IKE daemon procedure name.
#
# See the IP System Administrator's Commands book for more information
# about the modify command.
#
# See the IP Configuration Reference book for more information about
# the IkeConfig and NssConfig statements and their individual
# parameters.

IkeConfig
{
# IkeSyslogLevel    0-255             (dynamically modifiable)
# Specifies the level of logging to obtain from the IKE daemon.
# To specify a combination of log levels, add the level numbers.
# The supported levels are:
#   0 - IKE_SYSLOG_LEVEL_NONE       - Disable IKE daemon syslog messages
#   1 - IKE_SYSLOG_LEVEL_MINIMUM    - Minimal IKE daemon syslog output
#   2 - IKE_SYSLOG_LEVEL_SADETAIL   - Always output detailed Security
#                                     Association (SA) information when
#                                     available
#   4 - IKE_SYSLOG_LEVEL_DEBUGSA    - Include additional debug
#                                     information for SA negotiations
#   8 - IKE_SYSLOG_LEVEL_FMTPKTTRC  - Formatted IKE message trace
#  16 - IKE_SYSLOG_LEVEL_UNFPKTTRC  - Unformatted IKE message trace
#  32 - IKE_SYSLOG_LEVEL_VERBOSE    - Show cascaded error messages
#  64 - IKE_SYSLOG_LEVEL_CERTINFO   - Show certificates in CA cache when
#                                     cache is initially built or
#                                     rebuilt
# 128 - reserved
# Default:          1
  IkeSyslogLevel    1
```

```
# PagentSyslogLevel 0-255            (dynamically modifiable)
# Specifies the level of logging to obtain from pagent through the PAPI.
# To specify a combination of log levels, add the level numbers.
# The supported levels are:
#   1 - PAGENT_SYSLOG_LEVEL_EMERG   - A panic condition
#   2 - PAGENT_SYSLOG_LEVEL_ALERT   - Requires immediate action
#   4 - PAGENT_SYSLOG_LEVEL_CRIT    - Critical condition
#   8 - PAGENT_SYSLOG_LEVEL_ERR     - Error messages
#  16 - PAGENT_SYSLOG_LEVEL_WARNING - Warning messages
#  32 - PAGENT_SYSLOG_LEVEL_NOTICE  - Notice messages
#  64 - PAGENT_SYSLOG_LEVEL_INFO    - Informational messages
# 128 - PAGENT_SYSLOG_LEVEL_DEBUG   - Debug messages
# Default:          0
  PagentSyslogLevel 0

# Keyring           userid/ringname   (not dynamically modifiable)
# The owning userid and ringname used by the IKE server when performing
# RSA Signature Mode of authentication.  The userid must be the userid
# of the process under which IKE will run.
# Default:          iked/keyring
  Keyring           iked/keyring

# IkeRetries        1-8             (dynamically modifiable)
# Specifies the number of times that an unanswered IKE negotiation
# message is retransmitted before the negotiation is cancelled.
# Default:          6
  IkeRetries        6

# IkeInitWait       1-15             (dynamically modifiable)
# Specifies the number of seconds to wait before the first
# retransmission of an unanswered IKE message
# Default:          2
  IkeInitWait       2

# FIPS140           yes,no            (not dynamically modifiable)
# Specifies whether the IKE daemon should perform cryptographic
# operations by invoking cryptographic modules that are compliant with
# Federal Information Processing Standard (FIPS) publication 140-2's
# Level 1 security requirements.
# Default:          no
  FIPS140           no

# Echo              yes,no            (dynamically modifiable)
# Echoes all IKE daemon log messages to the job output file,
# specified by the IKEDOUT DD (JCL) statement.
# Default:          no
  Echo              no

# PagentWait        0-9999            (not dynamically modifiable)
# The time limit in seconds to wait for connection to the policy agent.
# A value of 0 means retry forever.
# Default:          0
  PagentWait        0

# SupportedCertAuth label             (dynamically modifiable)
# Specifies the label of a Certificate Authority(CA) certificate on the
# IKE server's keyring.  Use multiple instances of this keyword to
# specify multiple CA certificates.
# Default:          <none>

# NetworkSecurityServer address  Port 4159 Identity IpAddr 1.2.3.4
# Default: none                       #(dynamically modifiable)
# NetworkSecurityServerBackup address  Port 4159 Identity IpAddr 2.2.3.4
# Default: none                       #(dynamically modifiable)

# NssWaitLimit      1-300             (dynamically modifiable)
# Specifies the number of seconds that a Network Security client
# will wait between connection attempts when trying to establish a
# connection with a Network Security Server.
# Default: 60
  NssWaitLimit 60

# NssWaitRetries    1-10              (dynamically modifiable)
# Specifies the number of times that a Network Security client will
# attempt to establish a connection with the primary Network Security
# Server before attempting to establish a connection with the backup
# server.
# Default: 3
  NssWaitRetries  3
```

```
# SMF119            None, IKETunnel, DynTunnel, IKEAll  (dynamically
#                                                        modifiable)
# Specifies the level of logging to send to the SMF facility.
# IKEAll is equivalent to specifying SMF119 IKETunnel and
# SMF119 DynTunnel on two separate lines.
# The supported levels are:
# None              No SMF records
# IKETunnel         Phase 1 related SMF records
# DynTunnel         Phase 2 related SMF records
# IKEAll            Phase 1 and Phase 2 related SMF records
# Default:          None
  SMF119            None
}

# NssStackConfig    stackname         (dynamically modifiable)
# Used to configure a stack as a Network Security client.
# Use one NssStackConfig statement for each TCPIP stack that you wish
# to configure as a Network Security client. TCPIP stacks that do not
# have a corresponing NssStackConfig statement will be serviced by
# local IKE resources only.
#
# NssStackConfig    TCPCS
# {
# Clientname        clientname        (dynamically modifiable)
# Specifies the Network Security client name for the stack.  Client
# names for stacks typically have the form sysname_stackname, where
# sysname is the MVS system name, and stackname is the TCP/IP stack
# name.   This name must match the clientname portion of the associated
# SERVAUTH profiles:
#   - EZB.NSS.sysname.clientname.IPSEC.CERT
#   - EZB.NSS.sysname.clientname.IPSEC.NETMGMT
# The client name may be from 1 to 24 characters long.
# Default: <systemname>_<stackname>
# ClientName  MYSYSTEM_TCPCS
#
# ServiceType       RemoteMgmt, Cert  (dynamically modifiable)
# Specifies that the stack is requesting a type of centralized
# management via a Network Security Server.  This statement will occur
# once for each type of service that the stack is requesting. Supported
# service types are:
#    - RemoteMgmt
#    - Cert
# Defaults: None
# ServiceType RemoteMgmt
# ServiceType Cert
#
# Userid            userid            (dynamically modifiable)
# Specifies the RACF userid that will be used to verify access for this
# stack to the services provided by the Network Security Server. Userid
# may be from 1-8 characters long.
# Defaults: None.
# UserId SMITHXYZ
#
#Authby             Password password         (dynamically modifiable)
#                   Passticket                (dynamically modifiable)
# Specifies the mechanism by which the Network Security Server should
# authenticate the client TCPIP stack. Supported mechanisms are RACF
# password or RACF passticket.
#
#      Password  password
#        password is the RACF password for the userid specified for the
#        UserId.
#
#      Passticket
#        A RACF Passticket is generated for authorization.
#
# Default: none.  One (and only one) of Password or Passticket must be
#          specified.
# Authby Password secretxyz
# }
```

Figure 14. Sample IKE daemon configuration file

Chapter 9. Network security services server

This topic contains the following information about the Network security services (NSS) server:

- "Starting Network security services server using z/OS UNIX"
- "Network security services server cataloged procedure"
- "Network security services server environment variables" on page 461
- "Network security services server configuration file statements" on page 462

Starting Network security services server using z/OS UNIX

Start NSS server from the z/OS shell using the following syntax:

```
►►──nssd─────────────────────────────────────────────────────────────►◄
```

Tip: When you are starting the NSS server from the z/OS UNIX shell, set the environment variable _BPX_JOBNAME. This enables a specific job name to be used when reserving ports for the NSS server. You can also use this name with the STOP or MODIFY console commands. For more information about _BPX_JOBNAME, see z/OS UNIX System Services Planning.

Network security services server cataloged procedure

Update the cataloged procedure, NSSD, by copying the sample in SEZAINST(NSSD), to your system or recognized PROCLIB. Specify the NSS server parameters and change the data set names to suit your local configuration. See SEZAINST(EZARACF) for external security manager considerations for started procedures. After the NSSD procedure has been started, you can specify a different NSSD configuration file by using the Modify command with the FILE parameter. For example:

```
MODIFY NSSD,REFRESH,FILE='/etc/security/nssd.conf2'
```

```
//NSSD PROC
//*
//* IBM Communications Server for z/OS
//* SMP/E distribution name: EZBIMPRC
//*
//* 5650-ZOS Copyright IBM Corp. 2007, 2013
//* Licensed Materials - Property of IBM
//* "Restricted Materials of IBM"
//* Status = CSV2R1
//*
//*
//NSSD EXEC PGM=NSSD,REGION=0K,TIME=NOLIMIT,
// PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/'
//*
//* Provide environment variables to run with the desired
//* configuration. As an example, the data set or file specified by
//* STDENV could contain:
//*
//* NSSD_FILE=/etc/security/nssd.conf
//* NSSD_CTRACE_MEMBER=CTINSS01
//* NSSD_CODEPAGE=IBM-1047
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//* For information on the above environment variables, refer to the
//* IP Configuration Reference.
//*
//STDENV DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV DD DSN=TCPIP.NSSD.ENV(NSSD),DISP=SHR
//* Sample file containing environment variables:
//*STDENV DD PATH='/etc/security/nssd.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
```

Figure 15. NSSD cataloged procedure

Network security services server environment variables

Table 21 provides a list of environment variables used by the NSS server that you can alter for a particular installation.

Table 21. NSS server environment variables

Environment variable	Description	Any specific coding rules
NSSD_CODEPAGE	Used by the NSS server to specify the EBCDIC codepage to be used for the configuration file. The default codepage is IBM-1047.	The following codepages are supported: • IBM-037 • IBM-273 • IBM-274 • IBM-275 • IBM-277 • IBM-278 • IBM-280 • IBM-281 • IBM-282 • IBM-284 • IBM-285 • IBM-297 • IBM-500 • IBM-871 • IBM-1047 • IBM-1140 • IBM-1141 • IBM-1142 • IBM-1143 • IBM-1144 • IBM-1145 • IBM-1146 • IBM-1147 • IBM-1148 • IBM-1149 Example: `NSSD_CODEPAGE=IBM-1141`
NSSD_FILE	Used by the NSS server in the search order for the NSS server configuration file. For details on the search order used for locating this configuration file, see "TCP/IP configuration data sets" on page 1.	If this environment variable is not defined, the default value used by the NSS server is `/etc/security/nssd.conf`. Example: `NSSD_FILE=/etc/security/nssd.conf`
NSSD_PIDFILE	Used by the NSS server in the search order for the NSS server PID file. The search order for the NSS server PID file is as follows: 1. NSSD_PIDFILE environment variable 2. /etc/nssd.pid	If this environment variable is not defined, the default value used by the NSS server is /etc/nssd.pid. Example: `NSSD_PIDFILE=/etc/nssd.pid`

Table 21. NSS server environment variables (continued)

Environment variable	Description	Any specific coding rules
NSSD_CTRACE_MEMBER	Used by the NSS server to specify the name of a parmlib member that contains default CTRACE settings. The NSSD_CTRACE_MEMBER environment variable is read by the NSS server only during initialization. Changes to the NSSD_CTRACE_MEMBER after NSS server initialization have no effect.	If this environment variable is not defined, the default value used by the NSS server is CTINSS00. Example: `NSSD_CTRACE_MEMBER=CTINSS00`

Network security services server configuration file statements

The configuration of the NSS server contains parameters that define the behavior of the daemon. If any parameter is omitted from the configuration file, then default values are provided for parameters that support a default. If a configuration file is not specified, then default values are provided for all parameters.

If a configuration error is detected during startup, then the NSS server logs the error and exits. If a configuration error is detected during a dynamic refresh, then the entire refresh is rejected, the error is logged, and the NSS server continues running with the old configuration values.

The NSS server uses the following search order to locate the configuration file (highest priority is listed first):

1. The name of a file or MVS data set specified by the NSSD_FILE environment variable.
2. /etc/security/nssd.conf

Tip: You can use the IBM Configuration Assistant for z/OS Communications Server to establish NSS server settings. Establish the settings using the NSS perspective of the Configuration Assistant, and then click **Install Configuration File** on the **Image Information** tab to store the generated NSS server configuration file on the z/OS system.

NSS server configuration file sample

This topic shows the NSS server configuration file sample.

```
#
# IBM Communications Server for z/OS
# SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZANSCFG
#
# Licensed Materials - Property of IBM
# 5694-A01 Copyright IBM Corp. 2007, 2010
# Status = CSV1R12
#
# /etc/security/nssd.conf (network security services (NSS) server
#                          configuration)
#
# This file contains sample configuration parameters for the NSS server.
# The search order used by the NSS server to locate the initial
# configuration file is (highest priority listed first):
#
# 1) The name of a file or MVS data set specified by the NSSD_FILE
# environment variable.
# 2) /etc/security/nssd.conf
#
# Some parameters may be dynamically modified after the
# NSS server has been started. The parameters that are
# dynamically modifiable are noted below.
#
# To dynamically modify the NSS server's configuration parameters
# first edit the configuration file and then issue the modify command
# (while the NSS server is running). This causes the NSS server to
# re-read the configuration file.
#
# Example: MODIFY NSSD,REFRESH
# Note: NSSD is the NSS server procedure name.
#
# After the NSS server has been started, a different configuration
# file can be specifed by using the modify command with the FILE
# parameter. This allows modifiable parameters to be
# dynamically altered while the NSS server is running. Note that
# the parameter values modified in this fashion are not
# persistent. To make the changes persistent, edit the configuration
# file that is located at NSS server initialization time according
# to the search order described previously.
#
# Example: MODIFY NSSD,REFRESH,FILE='/etc/security/nssd.conf2'
# Note: NSSD is the NSS server procedure name.
#
# See the IP System Administrator's Commands book for more information
# about the modify command.
#
# See the IP Configuration Reference book for more information about
# the individual parameters.
#
# Blank lines, empty lines and lines beginning with the '#' char as the
# first non-space character are ignored.
# ------------
#
# The NssConfig statement contains parameters that apply globally
# across all supported disciplines.

NssConfig
{
# Port portNumber                               (dynamically modifiable)
# This is the TCP port to which the NSS server will bind.
# Default: 4159
Port 4159

# SyslogLevel 0-255                             (dynamically modifiable)
# Specifies the level of logging to obtain from the NSS server.
# To specify a combination of log levels, add the level numbers.
# The supported levels are:
# 0   - NSS_SYSLOG_LEVEL_NONE      - Disable NSS server syslog messages
# 1   - NSS_SYSLOG_LEVEL_MINIMUM   - Minimal NSS server syslog messages
# 2   - NSS_SYSLOG_LEVEL_VERBOSE   - Include cascaded internal error
#                                    messages (for IBM service)
# 4   - NSS_SYSLOG_LEVEL_CERTINFO  - Include info about certificate
#                                    cache
# 8   - NSS_SYSLOG_LEVEL_CLIENTLIFECYCLE - Include info about client
#                                          lifecycle
# 16  - NSS_SYSLOG_LEVEL_SAF_ACCESS_INFO - Include info about SAF
#                                          access operations
# 32  - reserved
# 64  - reserved
# 128 - reserved
# Default: 1
SyslogLevel 1

# KeyRing userid/ringname                       (dynamically modifiable)
# The NSS server attempts to open the configured key ring during
# startup. The key ring is used throughout the IPSec and XMLAppliance
# disciplines to locate certificates and/or private keys to be used for
# centralized cryptographic services.
# When using a key ring owned by the NSS server, specify only the
# ringname value. When using a key ring owned by another user, specify
# the ring name as a userid/ringname value.
# There is no default value. If KeyRing is not specified, then the
# NSS server cannot supply certificate services.
# KeyRing nssd/keyring

# Discipline disciplineName Enable | Disable    (dynamically modifiable)
# Specifies a discipline that is enabled or disabled by the NSS server.
# Supported disciplines are:
#   IPSec -
#     Includes the IPSec certificate service and IPSec network
#     management service. The default for the IPSec discipline is
#     'Enable'.
#   XMLAppliance -
#     Includes the XMLAppliance SAF access service, XMLAppliance private
#     key service, and XMLAppliance certificate service. The default for
#     the XMLAppliance discipline is 'Enable'.
# Default: IPSec Enable
# Default: XMLAppliance Enable
Discipline IPSec Enable
Discipline XMLAppliance Enable
}

# The IPSecDisciplineConfig statement contains parameters that apply
# to only the IPSec discipline.

IPSecDisciplineConfig
{
# FIPS140 Yes | No                                  (non-refreshable)
# Specifies whether the NSS server should perform cryptographic
# operations for the IPSec Discipline by invoking cryptographic modules
# that are compliant with Federal Information Processing Standard (FIPS)
# publication 140-2's Level 1 security requirements.
# Default: No
FIPS140 No

# URLCacheInterval minutes                     (dynamically modifiable)
# Specifies the maximum amount of time in minutes that data retrieved
# from an HTTP server will be cached before an attempt to reload the
# data is made. If minutes is specified as 0 then data retrieved from
# HTTP servers will not be cached.
# Default: 10080
URLCacheInterval 10080

# CertificateURL certlabel url                 (dynamically modifiable)
# Maps a certificate (specified by certlabel) to a URL which identifies
# a file on an HTTP server that contains a DER-encoded representation
# of the certificate.  Zero or more of these mappings may be specified.
# Note that this keyword only applies to network security clients that
# are using the IPSec certificate service in support of IKE version 2
# Phase 1 SA negotiations.
# There is no default value.
# CertificateURL CACert1 http://mycompany.com/ca_cert1.der

# CertificateBundleURL certlabel url           (dynamically modifiable)
# Maps a certificate (specified by certlabel) to a URL which identifies
# a file on an HTTP server that contains an x509 certificate bundle
# pertaining to the certificate.  Zero or more of these mappings may be
# specified.  Note that this keyword only applies to network security
# clients that are using the IPSec certificate service in support of IKE
# version 2 Phase 1 SA negotiations.
# There is no default value.
# CertificateBundleURL CACert2 http://mycompany.com/certbundle2.bndl
}
```

Figure 16. NSS server configuration file sample

IPSecDisciplineConfig statement

The IPSecDisciplineConfig statement contains parameters that apply to the IPSec discipline only. If more than one IPSecDisciplineConfig statement is coded, the last one is used. If a parameter within the IPSecDisciplineConfig statement is specified more than once, the value from the last one is used.

Syntax

```
►►──IPSecDisciplineConfig──┤ Braces &     Parms on Separate Lines ├────────────►◄
```

Braces & Parms on Separate Lines:

```
├──┬─{─────────────────────────────────┬────────────────────────────────────┤
   ├─┤ IPSecDisciplineConfig Parameters ├─┤
   └─}─────────────────────────────────┘
```

IPSecDisciplineConfig Parameters:

```
   ┌─FIPS140 no─────────┐ ┌─URLCacheInterval 10080─┐
├──┼────────────────────┼─┼────────────────────────┼──────────────────────►
   └─FIPS140 ──┬─yes─┬──┘ └─URLCacheInterval───────┘
               └─no──┘

    ┌──────────────────────────┐  ┌───────────────────────────────┐
    ▼                          │  ▼                               │
►──┬────────────────────────────┬──┬─────────────────────────────────┬─────┤
   └─CertificateURL label url─┘   └─CertificateBundleURL label url─┘
```

Parameters

`FIPS140 Yes | No`

Specifies whether the NSS server should perform cryptographic operations by invoking cryptographic modules that are designed to meet the Level 1 security requirements documented in the Federal Information Processing Standard (FIPS) publication 140 (FIPS 140).

yes	Perform all IPSec discipline cryptographic operations using cryptographic modules that are designed to meet FIPS 140 requirements. When the value of yes is specified, the NSS server is running in FIPS 140 mode.
no	NSS server might perform some cryptographic operations using cryptographic modules that do not adhere to the FIPS 140 requirements. When the value of no is specified, the NSS server is not running in FIPS 140 mode.

Requirement: ICSF must be active before starting the NSS server when FIPS140 YES is specified. For information about configuring ICSF to support FIPS 140-2, see Operating in compliance with FIPS 140-2 in z/OS Cryptographic Services ICSF Writing PKCS #11 Applications.

Rule: If the FIPS140 parameter is modified while the NSS server is running it will not take effect until the NSSD is restarted. Attempts to modify the value while the NSS server is running are ignored and a warning message is issued.

Tip: Enabling FIPS 140 mode provides a higher degree of assurance of the integrity of the cryptographic modules that the NSS server uses, including ICSF and System SSL. However, enabling FIPS 140 mode might require additional setup and configuration, it will restrict the available set of cryptographic algorithms, and it might result in a reduction in performance. See Cryptographic standards and FIPS 140 in z/OS Communications Server: IP Configuration Guide for more information.

`URLCacheInterval` *`minutes`*

Specifies the maximum amount of time in minutes that data retrieved from an HTTP server will be cached before an attempt to reload the data is made. If 0 is specified for the *minutes* value, then data retrieved from an HTTP server will not be cached. The default value is 10080 which is one week. The maximum value is 999999.

Tip: Table 22 shows when cached data must be reloaded.

Table 22. Cached data events that cause a reload

Cached data	Events that cause a reload
Certificate Data	The following events cause a reload: • The Validity notAfter time in the certificate is reached • The URLCacheInterval is reached • The NSSD MODIFY REFRESH command is issued
Certificate Bundle Data	The following events cause a reload: • The Validity notAfter time in any certificate in the bundle is reached • The nextUpdate time in any CRL in the bundle is reached • The URLCacheInterval is reached • The NSSD MODIFY REFRESH command is issued
Certificate Revocation Data	The following events cause a reload: • The nextUpdate time in the CRL is reached • The URLCacheInterval is reached • The NSSD MODIFY REFRESH command is issued

CertificateURL *label url*

The *label* is the label of a certificate on the key ring specified by the KeyRing parameter. If this label value contains imbedded blanks, then the value must be enclosed in double quote characters ("). Empty ("") and blank (" ") label names are not allowed. Any leading or trailing blanks within the double quotes will be ignored (for example, " label name " is treated as "label name"). If the string also contains a double quote character, then the imbedded double quote character must be coded as a sequence of two such characters (""). For example, the label in the following statement contains both imbedded blanks and imbedded double quotes:

```
CertificateURL   "my ""new"" certificate"    http://xyz.edu/cert51
```

The *url* is an HTTP based URL identifying a file on an HTTP server that contains the DER encoded representation of the certificate identified by *label*. The file should not contain the private key associated with the certificate. See Using hash and URL certificate encoding types in z/OS Communications Server: IP Configuration Guide for additional details.

Rule: If the same label is specified on multiple CertificateURL statements only the last CertificateURL statement for that label is used.

Tip: This keyword is applicable only to network security clients utilizing certificate services during an IKE version 2 Phase 1 SA negotiation.

CertificateBundleURL *label url*

The *label* value is the label of a certificate on the key ring specified by the

KeyRing parameter. If this label value contains imbedded blanks, then the value must be enclosed in double quote characters ("). Empty ("") and blank (" ") label names are not allowed. Any leading or trailing blanks within the double quotes will be ignored (for example, " label name " is treated as "label name"). If the string also contains a double quote character, then the imbedded double quote character must be coded as a sequence of two such characters (""). For example, the label in the following statement contains both imbedded blanks and imbedded double quotes:

```
CertificateBundleURL   "my ""new"" certificates"    http://xyz.edu/certbundle
```

The *url* is an HTTP based URL identifying a file on an HTTP server that contains an x509 certificate bundle pertaining to the certificate identified by *label*. The z/OS UNIX certbundle command may be used to create an x509 certificate bundle. See Using hash and URL certificate encoding types in z/OS Communications Server: IP Configuration Guide for additional details.

Rules:

- If the same label is specified on multiple CertificateBundleURL statements, only the last CertificateBundeURL statement for that label is used.
- If the same label is specified on both a CertificateURL statement and CertificateBundleURL statement, the statement specified last is used.

Tip: This keyword is applicable only to network security clients utilizing certificate services during an IKE version 2 Phase 1 SA negotiation.

NssConfig statement

The NssConfig statement contains parameters that apply globally to the NSS server and all supported disciplines. If more than one NssConfig statement is coded, the parameters coded within all the statements are combined as if they had all been coded under one NssConfig statement. If a parameter within the NssConfig statement is specified more than once, the value from the last one is used.

Syntax

```
►►──NssConfig──┤ Braces &    Parms on Separate Lines ├──────────────────────►◄
```

Braces & Parms on Separate Lines:

```
├───┬{───────────────────────┬──────────────────────────────────────────────┤
    ├┤ NSSConfig Parameters ├┤
    └}───────────────────────┘
```

NSSConfig Parameters:

```
    ┌Port 4159─┐    ┌SyslogLevel 0─┐
├───┴Port n────┴────┴SyslogLevel n─┴──┬──────────────────────┬──────────────►
                                      ├KeyRing userid/ringname┤
                                      └KeyRing ringname───────┘
```

```
     ┌──────────────────────────────────────────┐
     │ ┌─Discipline XMLAppliance Enable─────────┐ │
     ▼ ├─Discipline IPSec Enable───────────────┤ │
►──────┼────────────────────────────────────────┼─┴────────────────────────────┤
       └─Discipline──┬─IPSec────────┬──┬─Enable──┬┘
                     └─XMLApplicance┘  └─Disable─┘
```

Parameters

Port *n*

The TCP port that the NSS server binds to. All NSS clients must connect to the server through this port.

The default value is 4159. Valid values are in the range 1 - 65535. Use the MODIFY NSSD,REFRESH command to change the value of this parameter. When the TCP port is changed, existing connections remain open, but all new client connections must come through the new port.

Tip: The NSS server binds to INADDR_ANY. Configuring NSS clients to connect to the NSS server on a dynamic VIPA might increase availability of the NSS server. See NSS server failover considerations in z/OS Communications Server: IP Configuration Guide for more information.

SyslogLevel *level*

Specifies the level of logging to be obtained from the NSS server. The following levels are supported:

0 - NSS_SYSLOG_LEVEL_NONE
Disable NSS server syslog messages.

1 - NSS_SYSLOG_LEVEL_MINIMUM
Minimal NSS daemon syslog output.

2 - NSS_SYSLOG_LEVEL_VERBOSE
Include cascaded internal error messages (for IBM service).

4 - NSS_SYSLOG_LEVEL_CERTINFO
Include information about certificate cache.

8 - NSS_SYSLOG_LEVEL_CLIENTLIFECYCLE
Include information about client lifecycle (connect, update, and disconnect).

16 - NSS_SYSLOG_LEVEL_SAF_ACCESS_INFO
Include information about SAF access operations.

32	Reserved
64	Reserved
128	Reserved

These levels can be added together to create a cumulative logging effect.

Use the MODIFY NSSD,REFRESH command to change this value. The default value is 1.

Rules:
- The default SyslogLevel is in effect until the parameter is read from the configuration file.
- Any level higher than 1 automatically includes 1.

KeyRing *ringname* **|** *userid/ringname*

The owning user ID and ring name used by the NSS server when you are creating and verifying signatures on behalf of a NSS client. When using a key ring owned by the NSS server, specify the ring name as *ringname* value. When using a key ring owned by another user, specify the ring name as a *userid/ringname* value. There is no default value. If KeyRing is not specified, then the NSS server cannot supply certificate services.

Restriction: The NSS server does not support PKCS #11 Tokens for the KeyRing parameter.

Use the MODIFY NSSD,REFRESH command to change this value.

Discipline *discipline* **Enable | Disable**

Specifies that a discipline is enabled or disabled by the NSS server. Valid disciplines are:

IPSec Includes the IPSec certificate service and IPSec remote management service. The default for the IPSec discipline is Enable.

XMLAppliance
Includes the XMLAppliance SAF access service, the XMLAppliance certificate service, and the XMLAppliance private key service. The default for the XMLAppliance discipline is Enable.

Use the MODIFY NSSD, REFRESH command to change which disciplines are enabled or disabled, as follows:

Enabling a discipline

If, during refresh processing, the NSS server detects a Discipline statement that has been added or modified with the Enable keyword, the NSS server enables the required services to allow NSS clients to connect to the indicated discipline.

Disabling a discipline

If, after a refresh, a Discipline statement was modified with the Disable keyword, then connections for all NSS clients of the indicated discipline are removed and services for the indicated discipline are disabled. The NSS server prevents new clients from connecting to the indicated discipline.

Chapter 10. Defense Manager daemon

The Defense Manager daemon (DMD) is an integral part of defensive filtering. The z/OS UNIX **ipsec** command provides the user interface to add, update, delete, and display defensive filters. The DMD sits between the z/OS UNIX **ipsec** command and the TCP/IP stacks. The DMD manages the installation of defensive filters into the TCP/IP stacks. One instance of the DMD manages all stacks on the z/OS image. The DMD must be active for defensive filters to be added, updated, or deleted. For more information about defensive filtering, see z/OS Communications Server: IP Configuration Guide. For more information about the z/OS UNIX ipsec command, see z/OS Communications Server: IP System Administrator's Commands.

This topic contains the following information about the DMD:
- "Starting the DMD using z/OS UNIX (optional)"
- "The Defense Manager daemon cataloged procedure (optional)"
- "DMD environment variables" on page 473
- "DMD configuration file statements" on page 474

Starting the DMD using z/OS UNIX (optional)

If the DMD is to be started from the z/OS UNIX shell, use the following syntax:

```
►►──dmd──────────────────────────────────────────────────────────►◄
```

Tip: When you are starting the DMD from the z/OS UNIX shell, set the environment variable _BPX_JOBNAME so that you can use a specific job name with the STOP or MODIFY console commands. For more information about _BPX_JOBNAME, see z/OS UNIX System Services Planning.

The Defense Manager daemon cataloged procedure (optional)

If the DMD is to be started by a procedure, update the cataloged procedure, DMD, by copying the sample in SEZAINST(DMD) to your system or recognized PROCLIB. Specify the DMD parameters and change the data set names that are appropriate for your local configuration. See SEZAINST(EZARACF) for external security manager considerations for started procedures. After you have started the DMD procedure, you can specify a different DMD configuration file by using the MODIFY command with the FILE parameter. For example:

```
MODIFY DMD,REFRESH,FILE='/etc/security/dm.conf2'
```

```
//DMD PROC
//*
//* IBM Communications Server for z/OS
//* SMP/E distribution name: EZADMD
//*
//* 5650-ZOS Copyright IBM Corp. 2008, 2013
//* Licensed Materials - Property of IBM
//* Status = CSV2R1
//*
//*
//DMD EXEC PGM=DMD,REGION=0K,TIME=NOLIMIT,
// PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/'
//*
//* Provide environment variables to run with the desired
//* configuration. As an example, the data set or file specified by
//* STDENV could contain:
//*
//* DMD_FILE=/etc/security/dmd.conf
//* DMD_CTRACE_MEMBER=CTIDMD00
//* DMD_PIDFILE=/var/dm/dmd.pid
//* DMD_CODEPAGE=IBM-1047
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//* For information on the above environment variables, refer to the
//* IP Configuration Reference.
//*
//STDENV DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV DD DSN=TCPIP.DMD.ENV(DMD),DISP=SHR
//* Sample file containing environment variables:
//*STDENV DD PATH='/etc/security/dmd.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
```

Figure 17. DMD cataloged procedure

DMD environment variables

Table 23 provides a list of environment variables used by the DMD that can be tailored to a particular installation.

Table 23. Defense Manager daemon (DMD) environment variables

Environment variable	Description	Any specific coding rules
DMD_CODEPAGE	Used by the DMD to specify the EBCDIC code page to be used for the configuration file. The default code page is IBM-1047.	The following code pages are supported: • IBM-037 • IBM-273 • IBM-274 • IBM-275 • IBM-277 • IBM-278 • IBM-280 • IBM-281 • IBM-282 • IBM-284 • IBM-285 • IBM-297 • IBM-500 • IBM-871 • IBM-1047 • IBM-1140 • IBM-1141 • IBM-1142 • IBM-1143 • IBM-1144 • IBM-1145 • IBM-1146 • IBM-1147 • IBM-1148 • IBM-1149 Example: `DMD_CODEPAGE=IBM-1141`
DMD_FILE	Used by the DMD in the search order for the DMD configuration file. For details about the search order used for locating this configuration file, see "TCP/IP configuration data sets" on page 1.	Example: `DMD_FILE=/etc/security/dmd.conf`
DMD_PIDFILE	Used by the DMD in the search order for the DMD PID file. The search order for the DMD PID file is as follows: 1. DMD_PIDFILE environment variable 2. /var/dm/dmd.pid	Example: `DMD_PIDFILE=/var/dm/dmd.pid`

Table 23. Defense Manager daemon (DMD) environment variables (continued)

Environment variable	**Description**	**Any specific coding rules**
DMD_CTRACE_MEMBER	Used by the DMD to specify the name of a parmlib member that contains default CTRACE settings. The DMD_CTRACE_MEMBER environment variable is read by the DMD only during initialization. Changes to the DMD_CTRACE_MEMBER after DMD initialization have no effect.	If not defined, the default value used by the DMD is CTIDMD00. Example: `DMD_CTRACE_MEMBER=CTIDMD00`

DMD configuration file statements

The DMD configuration file contains parameters that define the behavior of the daemon. These parameters are contained in two statement types, DMConfig and DmStackConfig.

If a configuration error is detected during startup, the DMD logs the error and exits. If a configuration error is detected during a dynamic refresh, the entire refresh is rejected, the error is logged, and the DMD continues to run with the old configuration values.

The DMD uses the following search order to locate the configuration file:

1. The name of a file or MVS data set specified by the DMD_FILE environment variable
2. /etc/security/dmd.conf

All DMD configuration file statements are optional. An empty configuration file is permitted, but if no DmStackConfig statements are defined, no stacks are supported.

DmConfig statement

This statement contains configuration information for the DMD. Only one instance of the DmConfig statement can be included in the configuration file. If there are multiple instances of the DmConfig statement, an error is generated.

Syntax

```
►►──DmConfig──┤ Braces &     Parms on Separate Lines ├──────────────────────────►◄
```

Braces & Parms on Separate Lines:

```
├──┬─{─────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ DmConfig Parameters ├─┤
   └─}─────────────────────────┘
```

DmConfig Parameters:

```
   ┌SyslogLevel 7┐   ┌DefensiveFilterDirectory /var/dm/filters┐
├──┼─────────────┼───┼────────────────────────────────────────┼──}──────────┤
   └SyslogLevel n┘   └DefensiveFilterDirectory directory──────┘
```

Parameters

`SyslogLevel` *n*

Specifies the level of logging to obtain from the Defense Manager daemon. The following levels are supported:

0 - DM_SYSLOG_LEVEL_NONE
Disable Defense Manager syslog messages.

1 - DM_SYSLOG_LEVEL_MINIMUM
Minimal Defense Manager syslog output.

2 - DM_SYSLOG_LEVEL_LIFECYCLE_CLIENT
Include information about client connections and disconnections.

4 - DM_SYSLOG_LEVEL_LIFECYCLE_STACK
Include information about the cycling of stacks and the installation, deletion, or modification of defensive filters in a stack.

8 - DM_SYSLOG_LEVEL_VERBOSE
Include cascaded internal error messages (for IBM service).

16	Reserved
32	Reserved
64	Reserved
128	Reserved

You can add these levels to create a cumulative logging effect.

Use the MODIFY DMD,REFRESH command to change the SyslogLevel value.

The default value is 7.

Rules:

- The default SyslogLevel value is in effect until the parameter is read from the configuration file.
- Any level higher than 1 automatically includes 1.

`DefensiveFilterDirectory` *dirname*

The name of the directory in which the DMD creates a file for each stack using a copy of that stack's active defensive filters. These are binary files managed by the DMD; you must not manually modify them. This directory must exist when the DMD starts, and the DMD must have authority to create, delete, read, and write files in this directory.

This is not a refreshable parameter. Any refresh attempt fails if the new DefensiveFilterDirectory parameter value differs from the value that was used at server startup.

The default value is /var/dm/filters.

Rules:

- The binary files that DMD creates are persistent. If the DMD is restarted, the files are expected to reflect the active defensive filters in the TCP/IP stacks.

- Each stack can have a file that contains the persisted form of its active defensive filters. This file, if present, is named active.*stackname*. The name of the file that contains global defensive filters is active._globals_. This is a binary file managed by the DMD; do not manually modify it.
- When the DMD starts, each active.*stackname* file in the defensive filter directory is checked both for internal consistency and for consistency with the installed defensive filters in its corresponding stack (if that stack is active). If an inconsistency is detected, the file is considered to be corrupted or untrustworthy and message EZD1731I is written to syslog.
- When a defensive filter file is found to be untrustworthy, it is renamed by the DMD from active.*stackname* to untrusted.*stackname.tttttttt*, where *tttttttt* is the hexadecimal value of the current system timestamp as reported by the LE time() function. If the stack is active, any defensive filters currently in the stack are individually unaddressable by the DMD and can be referenced only by **ipsec** commands that accept or imply the ALL notation for addressing a stack's defensive filters.
- When defensive filtering is set to inactive for a stack, and if there is an active.*stackname* file for the stack, the file is renamed by the DMD from active.*stackname* to inactive.*stackname.tttttttt* where *tttttttt* is the hexadecimal value of the current system timestamp as reported by the language environment time() function. You can set defensive filtering to inactive by changing the mode of the stack on the DmStackConfig statement or by issuing the MODIFY FORCE_INACTIVE command.

DmStackConfig statement

This statement contains the Defense Manager daemon configuration information for a single TCP/IP stack.

Syntax

```
►►──DmStackConfig──stackname──┤ Braces & Parms on Separate Lines ├──────────►◄
```

Braces & Parms on Separate Lines:

```
├──┬─{──────────────────────────────────────┬─────────────────────────────┤
   │ ┌─Mode Active─────────┐                │
   ├─┴─Mode──┬─Active───┬──┴────────────────┤
   │         ├─Simulate─┤                   │
   │         └─Inactive─┘                   │
   │ ┌─MaxLifetime 1440─────┐               │
   ├─┴─MaxLifetime lifetime─┴───────────────┤
   │ ┌─DefaultLogLimit 0─────┐              │
   ├─┴─DefaultLogLimit──┬─0─┬┴──────────────┤
   │                    └─n─┘               │
   │ ┌──────────────────────────────────────┐
   │ ▼                                      │
   ├───┬─────────────────────────────────┬──┤
   │   └─Exclude──┬─ipaddress─────────────┤ │
   │              └─ipaddress/prefixLength┘ │
   └─}──────────────────────────────────────┘
```

Parameters

stackname
The name of the TCP/IP stack that is being configured for defensive filter support. This is a required parameter, and there is no default value.

Mode Active | Simulate | Inactive
Specifies the defensive filter mode for the TCP/IP stack. Possible values are:

Active When the stack specified by the *stackname* value is active and configured for IP security, it is managed by the DMD. Each defensive filter applied to that stack operates in the mode specified for the individual defensive filter, either block or simulate. Blocking mode discards packets that match the defensive filter. Simulate mode simulates a block for packets that match the defensive filter. When a packet matches a defensive filter with a simulate mode, a message is logged to indicate that the packet would have been discarded. However, the packet is not discarded and processing continues with IP filtering. For more information about simulate block behavior, see the z/OS Communications Server: IP Configuration Guide. This is the default.

Simulate
When the stack specified by the *stackname* value is active and configured for IP security, it is managed by the DMD. All defensive filters applied to that stack operate in simulate mode, overriding the mode specified for the individual filters. Simulate mode simulates a block. When a packet matches a defensive filter and the mode is simulate, a message is logged to indicate that the packet would have been discarded. However, the packet is not discarded and processing continues with IP filtering. For more information about simulate block behavior, see the z/OS Communications Server: IP Configuration Guide.

Tip: Simulate mode would typically be used in a test environment.

Inactive
If the stack specified by the *stackname* value is active and configured for IP security when the DMD starts, all defensive filters are removed from that stack and also from the DMD memory. No new defensive filters are installed in the stack while the mode is set to Inactive.

Tip: Use inactive mode to disable defensive filtering for the stack. If you remove the DmStackConfig statement for the stack from the DMD configuration file, the defensive filters currently installed in the stack are not removed. Without the DmStackConfig statement, you cannot use the z/OS UNIX **ipsec** command to delete defensive filters from the stack.

Use the MODIFY DMD,REFRESH command to change this value. You can also use the MODIFY DMD,FORCE_INACTIVE,*stackname* command to change the mode to Inactive without refreshing the configuration.

Exclude
Specifies an IP address or subnet to exclude from the effects of defensive filters installed in the stack specified by the *stackname* value. Inbound packets originating from an IP address in the exclusion list are excluded from defensive filter processing. Outbound packets destined to an IP address in the exclusion list are excluded from defensive filter processing.

Tip: Defensive filters are checked before IP security filters. To ensure that an administrator is not blocked by a defensive filter, you can exclude the administrator's IP address from defensive filter processing by specifying the administrator's address on the Exclude statement.

ipaddress
: Specifies a single IP address to be excluded from the effects of defensive filters. This value can be an IPv4 or IPv6 address.

ipaddress/prefixLength
: Specifies a prefix address specification that indicates the applicable IP addresses to be excluded from the effects of defensive filters. The *prefixLength* value is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP address matches this exclusion if its unmasked bits are identical to the defined unmasked bits.

There is a limit of 10 Exclude keywords on the DmStackConfig statement.

Use the MODIFY DMD,REFRESH command to change this value. In case of a successful refresh, the new list of exclusion addresses completely replaces the prior list of exclusion addresses.

This is an optional parameter, and there is no default value.

MaxLifetime
Specifies the maximum lifetime of a defensive filter in minutes. This value limits a defensive filter's lifetime when the defensive filter is first added or later updated. Lifetime values that exceed the MaxLifetime value are truncated to MaxLifetime minutes. Existing filters are not affected by a change to the MaxLifetime value that results from a MODIFY DMD,REFRESH operation.

lifetime Specifies the maximum number of minutes that are allowed for a defensive filter's lifetime. Valid values are in the range 1 - 20160 (2 weeks). The default is 1440 (1 day).

DefaultLogLimit
Specifies the default log limit for defensive filters that are added to this TCP/IP stack. When a defensive filter is added and the loglimit parameter is not specified on the z/OS UNIX ipsec add command, the DefaultLogLimit value will be used. The log limit value is used to enable or disable the limiting of defensive filter match messages (EZD1721I and EZD1722I). See filter-match logging in z/OS Communications Server: IP Configuration Guide for more information.

0 Disables the limiting of defensive filter match messages. If logging is being done for this defensive filter, a message is generated for each packet that matches the defensive filter. 0 is the default.

n Enables the limiting of defensive filter match messages. Valid values are in the range 1 - 9999. The value limits the average rate of filter-match messages generated in a 5-minute interval for a defensive filter. For example, a value of 100 limits the average rate of filter-match messages to 100 messages per 5-minute interval. A burst of up to 100 messages is allowed while maintaining the long-term average of 100 messages per 5-minute interval.

Result: The DMD installs and manages defensive filters only in TCP/IP stacks that are configured with a DmStackConfig statement in the DMD configuration file.

DMD configuration file sample

This topic shows the Defense Manager daemon configuration file sample.

```
#
# IBM Communications Server for z/OS
# SMP/E distribution path: /usr/lpp/tcpip/samples/dmd.conf
#
# 5650-ZOS Copyright IBM Corp. 2013.
# Licensed Materials - Property of IBM
# Status = CSV2R1
#
# /etc/security/dmd.conf (Defense Manager daemon configuration)
#
# This file contains sample Defense Manager daemon configuration
# parameters. The search order used by the Defense Manager daemon to
# locate the initial configuration file is (highest priority listed
# first):
#
# 1) The name of a z/OS UNIX file or z/OS dataset specified by the
#    DMD_FILE environment variable.
# 2) /etc/security/dmd.conf
#
# Some parameters are dynamically modifiable after the
# Defense Manager daemon has been started. The parameters that are
# dynamically modifiable are noted below.
#
# One way of dynamically modifying parameters is to edit
# the Defense Manager daemon configuration file after the Defense
# Manager daemon has been started and then issue a MODIFY command
# to cause the Defense Manager daemon to re-read the configuration file.
#
# Example: MODIFY DMD,REFRESH
# Note: DMD is the Defense Manager daemon procedure name.
#
# After the Defense Manager daemon has been started, a different
# configuration file can be specifed by using the Modify command with
# the FILE parameter. This allows modifiable parameters to be
# dynamically altered while the Defense Manager daemon is running. Note
# that the parameter values modified in this fashion are not
# persistent. To make the changes persistent, edit the dmd.conf
# file that is located at the Defense Manager daemon initialization
# time according to the search order described previously.
#
# Example: MODIFY DMD,REFRESH,FILE='/etc/security/dmd.conf2'
# Note: DMD is the Defense Manager daemon procedure name.
#
# See the IP System Administrator's Commands book for more information
# about the modify command.
#
# See the IP Configuration Reference book for more information about
# the individual parameters.
#
# Blank lines, empty lines and lines beginning with the '#' char as the
# first non-space character are ignored.
# ------------
#
DMConfig
{
# SyslogLevel 0-255 (dynamically modifiable)
#
# Specifies the level of logging to obtain from the Defense Manager
# daemon. To specify a combination of log levels, add the level numbers.
# The supported levels are:
# 0 - DM_SYSLOG_LEVEL_NONE - Disable the Defense Manager daemon syslog
#      messages
# 1 - DM_SYSLOG_LEVEL_MINIMUM - Minimal Defense Manager daemon syslog
#      output
# 2 - DM_SYSLOG_LEVEL_LIFECYCLE_CLIENT - Include info about the connect
#      and disconnect of clients.
# 4 - DM_SYSLOG_LEVEL_LIFECYCLE_STACK - Include info about the cycling
#      of stacks and the installation, deletion or modification of
#      defensive filters to the stack.
# 8 - DM_SYSLOG_LEVEL_VERBOSE - Include cascaded internal error messages
#      (for IBM service)
# 16 - reserved
# 32 - reserved
# 64 - reserved
# 128 - reserved
# Default: 7
```

```
SyslogLevel 7
#
# DefensiveFilterDirectory dirname (not dynamically modifiable)
#
# The name of the directory where the Defense Manager daemon will
# create a file for each stack with a copy of that stack's
# active defensive filters.  These are binary files managed
# by the Defense Manager daemon and must not be manually modified.
# This directory must exist when the Defense Manager daemon starts and
# the Defense Manager daemon must have authority to create, delete,
# read and write files in this directory.
#
# This is not a refreshable parameter. Any REFRESH attempt
# will fail if the new value of the DefensiveFilterDirectory
# parameter differs from the value used at server startup.
# Default: /var/dm/filters
DefensiveFilterDirectory /var/dm/filters
}
#DmStackConfig TCPCS
#{
# Mode  Active|Simulate|Inactive  (dynamically modifiable)
#
# This specifies the defensive filter mode for the TCP/IP stack.
#
# Valid options are:
#
#  Active    When stackname is active and configured for IP security,
#            it will be managed by the DMD.  Each defensive filter
#            applied to stackname will operate in the mode specified
#            for the individual defensive filter, block or simulate.
#            Blocking mode will discard packets that match the defensive
#            filter. Simulate mode will simulate a block for packets
#            that match the defensive filter.  When a packet matches a
#            defensive filter with a mode of simulate, a message will
#            be logged indicating that the packet would have been
#            discarded.  However, the packet will not be discarded and
#            IP filtering will continue.  For more information on
#            simulate block behavior see the IP Configuration Guide.
#   Simulate  When stackname is active and configured for IP security,
#             it will be managed by the DMD.  All defensive filters
#             applied to stackname will operate in simulate mode,
#             overriding the mode specified for the individual filters.
#             Simulate mode simulates a block.  When a packet matches a
#             defensive filter and the mode is simulate, a message will
#             be logged indicating that the packet would have been
#             discarded.  However, the packet will not be discarded
#             and IP filtering will continue.  For more information on
#             simulate block behavior see the IP Configuration Guide.
#             Tip: Simulate mode would typically be used in a test
#                  environment.
#   Inactive  If stackname is active and configured for IP security
#             when the DMD starts, then all defensive filters will be
#             removed from stackname and also from the DMD memory. No
#             new defensive filters will be installed in stackname
#             while the mode is Inactive.
#             Tip: Use Inactive mode to disable defensive filtering for
#                  stackname. Removing the DmStackConfig statement for
#                  stackname from the DMD configuration file does not
#                  remove defensive filters currently installed in
#                  stackname, and without the DmStackConfig statement,
#                  the z/OS UNIX ipsec command cannot delete defensive
#                  filters from stackname.
# Default: Active
# Mode Active
#
# MaxLifetime  lifetime  (dynamically modifiable)
#
# Valid values are 1-20160 minutes.
#
# Maximum number of minutes allowed for a defensive filter's lifetime.
# This value limits a defensive filter's lifetime when the defensive
# filter is first added or later updated.  Lifetime values that exceed
# MaxLifetime are truncated to MaxLifetime minutes.  Existing filters
# are not affected by a change to MaxLifetime resulting from a
# MODIFY DMD,REFRESH operation.
#
# Default: 1440 minutes (one day)
# MaxLifetime 1440
#
# DefaultLogLimit  loglimit  (dynamically modifiable)
```

```
#
# Valid values are 0-9999.
#
# Default log limit for defensive filters added to this TCP/IP stack.
# When a defensive filter is added and the loglimit parameter is not
# specified on the add command, the DefaultLogLimit value will be used.
# The log limit value is used to enable or disable limiting of
# defensive filter match messages (EZD1721I and EZD1722I).
#
# Valid values are:
#
#  0 - Disables limiting of defensive filter match messages. If logging
#      is being done for this defensive filter, a message is generated
#      for each packet that matches the defensive filter.
#  1-9999 - Enables limiting of defensive filter match messages. The
#           value limits the average rate of defensive filter match
#           messages generated in a 5-minute interval. For example, a
#           value of 100 would limit the average rate of defensive
#           filter match messages to 100 messages per 5 minutes,
#           allowing up to 100 messages to be issued in a burst, while
#           maintaining the long-term average.
#
# Default: 0
# DefaultLogLimit 0
#
# Exclude ipaddress | ipaddress/prefixLength (dynamically modifiable)
#
# Specifies an IP address or subnet to exclude from the effects of
# defensive filters. The ipaddress can be an IPv4 address or an IPv6
# address. Hostnames are not supported.
#
# There is a limit of 10 Exclude keywords on the DmStackConfig
# statement.
#
# In the case of a successful REFRESH, the new list of exclusion
# addresses will completely replace the prior list of exclusion
# addresses.
#
# Default: None.
#Exclude 9.29.4.25
#Exclude 9.29.4.26
#}
```

Figure 18. DMD configuration file sample

Chapter 11. OMPROUTE

This topic includes the following information:

- "Starting OMPROUTE using z/OS UNIX (optional)"
- "OMPROUTE cataloged procedure (optional)"
- "OMPROUTE parameters" on page 484
- "OMPROUTE environment variables" on page 485
- "OMPROUTE configuration file statements" on page 487

Starting OMPROUTE using z/OS UNIX (optional)

If OMPROUTE is to be started from the z/OS shell, use the following syntax:

```
►►──omproute──┬─────┬──┬─────┬──┬─────┬──┬──────┬──┬──────┬──────────────►◄
              └─-tn─┘  └─-dn─┘  └─-sn─┘  └─-6tn─┘  └─-6dn─┘
```

OMPROUTE cataloged procedure (optional)

If OMPROUTE is to be started by a procedure, update the cataloged procedure OMPROUTE by copying the sample in SEZAINST(OMPROUTE) to your system or recognized PROCLIB. Specify OMPROUTE parameters and change the data set names to suit your local configuration.

```
//*
//* TCP/IP for MVS
//* SMP/E Distribution Name: EZBORPRC
//*
//*        5650-ZOS Copyright IBM Corp. 1998, 2013
//*        Licensed Materials - Property of IBM
//*        This product contains "Restricted Materials of IBM"
//*        All rights reserved.
//*        US Government Users Restricted Rights -
//*        Use, duplication or disclosure restricted by
//*        GSA ADP Schedule Contract with IBM Corp.
//*        See IBM Copyright Instructions.
//*
//OMPROUTE PROC
//OMPROUTE EXEC PGM=OMPROUTE,REGION=10M,TIME=NOLIMIT,
// PARM=('ENVAR("_CEE_ENVFILE_S=DD:STDENV")/')
//*
//* Example of start parameters to OMPROUTE:
//*
//* PARM=('ENVAR("_CEE_ENVFILE_S=DD:STDENV")/-t1 -6t1')
//*
//*        Provide environment variables to run with the
//*        desired stack and configuration.  As an example,
//*        the file specified by STDENV could have these
//*        lines in it:
//*
//*           RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*           OMPROUTE_FILE=/u/usernnn/config.tcpcs2
//*           OMPROUTE_DEBUG_FILE=/tmp/logs/omproute.debug
//*           OMPROUTE_IPV6_DEBUG_FILE=/tmp/logs/omprout6.debug
//*           OMPROUTE_DEBUG_FILE_CONTROL=1000,5
//*
//*        For information on the above environment variables,
//*        refer to the IP CONFIGURATION GUIDE.
```

```
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the semicolon sign, ;, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=;
//*
//STDENV   DD PATH='/u/usernnn/envcs2',
//         PATHOPTS=(ORDONLY)
//*
//*        The stdout stream may be redirected to a HFS file as
//*        shown below.
//*        The PATHOPTS OTRUNC option will clear the stdout file
//*        every time OMPROUTE is started.  If you want to retain
//*        previous stdout information, change it to OAPPEND.
//*
//SYSPRINT DD SYSOUT=*
//*SYSPRINT DD PATH='/tmp/omproute.stdout',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//*        The stderr stream may be redirected to a HFS file as
//*        shown below.
//*        The PATHOPTS OTRUNC option will clear the stderr file
//*        every time OMPROUTE is started.  If you want to retain
//*        previous stderr information, change it to OAPPEND.
//*
//SYSOUT   DD SYSOUT=*
//*SYSOUT   DD PATH='/tmp/omproute.stderr',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Figure 19. OMPROUTE cataloged procedure

Restriction: When using _CEE_ENVFILE with an MVS data set, the data set must be allocated with RECFM=V. To use a RECFM=F data set, _CEE_ENVFILE_S should be used to prevent the environment variable values from being padded with blanks.

OMPROUTE parameters

-tn

External tracing level for OMPROUTE initialization and IPv4 routing protocols, where *n* is a supported trace level. The following values are supported:

1. Informational messages
2. Formatted packet trace and informational messages

-dn

Internal debugging level for OMPROUTE initialization and IPv4 routing protocols, where *n* is a supported debug level. This parameter is intended for service, as it provides information needed for debugging problems.

-sn
: Internal subagent debugging level, where *n* is a supported debug level. This parameter is intended for service, as it provides information needed for debugging problems.

-6tn
: External tracing level for IPv6 routing protocols, where *n* is a supported trace level. The following values are supported:

  1. Informational messages
  2. Formatted packet trace and informational messages

-6dn
: Internal debugging level for IPv6 routing protocols, where *n* is a supported debug level. This parameter is intended for service, as it provides information needed for debugging problems.

For more information about the *-dn*, *-6dn*, and *-sn* parameters, see z/OS Communications Server: IP Diagnosis Guide.

IBM Health Checker for z/OS can be used to check whether the total number of indirect routes in a TCP/IP stack routing table exceeds a maximum threshold. When this threshold is exceeded, OMPROUTE and the TCP/IP stack may potentially experience high CPU consumption from routing changes. A large routing table is considered to be inefficient in network design and operation. For more details about IBM Health Checker for z/OS, see z/OS Communications Server: IP Diagnosis Guide and IBM Health Checker for z/OS: User's Guide .

OMPROUTE environment variables

Table 24 provides a list of environment variables used by OMPROUTE and that be tailored to a particular installation:

Table 24. OMPROUTE environment variables

Environment variable	Description	Any specific coding rules (or a link to syntax)
RESOLVER_CONFIG	The RESOLVER_CONFIG variable is used by OMPROUTE to locate the resolver configuration file.	For more information about OMPROUTE's use of the resolver configuration file, see z/OS Communications Server: IP Configuration Guide. For more information about the RESOLVER_CONFIG environment variable, see z/OS UNIX System Services Planning.
OMPROUTE_CTRACE_MEMBER	The OMPROUTE_CTRACE_MEMBER variable is used by OMPROUTE to specify the name of the parmlib member containing CTRACE default settings. Use this environment variable to set different CTRACE options and buffer sizes for multiple OMPROUTE instances.	If not defined, the default value is CTIORA00.

Table 24. OMPROUTE environment variables (continued)

Environment variable	Description	Any specific coding rules (or a link to syntax)
OMPROUTE_DEBUG_FILE	The OMPROUTE_DEBUG_FILE variable is used by OMPROUTE to override the debug output destination for IPv4 dynamic routing protocols and for processing common to both IPv4 and IPv6 routing protocols. For more information about using this environment variable, see z/OS Communications Server: IP Configuration Guide.	**Restriction:** Ensure that the two debug file names are not identical in the characters up to the first period (.). This prevents problems when the initial debug files fill up and OMPROUTE tries to rename them, using the name up to the first period (.) with a sequence number substituted for the rest of the name.For more information about using this environment variable, see z/OS Communications Server: IP Configuration Guide.
OMPROUTE_IPV6_DEBUG_FILE	The OMPROUTE_IPV6_DEBUG_FILE variable is used by OMPROUTE to override the debug output destination for IPv6 routing protocols.	**Restriction:** Ensure that the two debug file names are not identical in the characters up to the first period (.). This prevents problems when the initial debug files fill up and OMPROUTE tries to rename them, using the name up to the first period (.) with a sequence number substituted for the rest of the name.For more information about using this environment variable, see z/OS Communications Server: IP Configuration Guide.
OMPROUTE_DEBUG_FILE_ CONTROL	The OMPROUTE_DEBUG_FILE_ CONTROL variable is used to specify the size and quantity of the files produced as a result of the OMPROUTE_DEBUG_FILE and OMPROUTE_IPV6_DEBUG_FILE environment variable.	For more information about using this environment variable, see z/OS Communications Server: IP Configuration Guide.
OMPROUTE_FILE	Used by OMPROUTE in the search order for the OMPROUTE configuration file. It uses the value as the name of an MVS data set or z/OS UNIX file to access the configuration data.	For more information about using this environment variable, see z/OS Communications Server: IP Configuration Guide.
OMPROUTE_OPTIONS	The OMPROUTE_OPTIONS variable is ignored and will be retired in a future release.	The only supported parameter is hello_hi, which is ignored and will be retired in a future release. The behavior, which is previously provided only when hello_hi is coded, is now always enabled. The hello_hi option is used to enforce OMPROUTE to process the hello packets at a higher priority than other updates. The hello_hi option also helps prevent adjacencies from failing when OMPROUTE is being flooded with protocol packets.

Table 24. OMPROUTE environment variables (continued)

Environment variable	Description	Any specific coding rules (or a link to syntax)
SNMP_PORT	Specifies the port to which a DPI subagent directs a connection query. The default is 161 (the default port on which the SNMP agent listens for queries).	None
TMPDIR	Holds the name of a directory where shell commands are free to create temporary working files. If TMPDIR is not defined, the default directory is /tmp.	For more information about using this environment variable, see z/OS UNIX System Services Command Reference.

OMPROUTE configuration file statements

Statements in the OMPROUTE configuration file have the following syntax:

```
type tag=value tag=value.. .. ..;
where:
type         Specifies what is to be configured
tag=value    Specifies a parameter and its associated value.
type=value   Used for statements that have only a single parameter.
```

Rules: The following list shows the syntax rules for the OMPROUTE configuration statements:

- Types, tags, and values can be specified in mixed case.
- Every configuration statement, with the exception of the INCLUDE statement, must end with a semicolon (;).
- Blanks and comments are supported. Comments are identified by a semicolon in any column. Comments cannot appear within a configuration statement.
- Statements can begin in any column.
- There must be no sequence numbers in the data set or file.
- Statements with only optional operands must have at least one operand coded, even if all operands have defaults.
- You can use static system symbols in OMPROUTE configuration file statements.

A sample OMPROUTE configuration file is provided in SEZAINST(EZAORCFG).

INCLUDE statement

This statement causes configuration statements from the specified data set to be included at the point at which the INCLUDE statement is encountered in the configuration file.

Rules:

- The INCLUDE statement must be the only configuration statement on the line.
- The INCLUDE statement must not end with semicolon.
- There must be no more than 10 nested INCLUDE statements.
- You can specify static system symbols as part of the data set name.

Syntax

```
►►──INCLUDE──┬─//'fully qualified MVS dataset name'─┬──────────────────────────►◄
             └─/file system absolute pathname───────┘
```

Parameters

//'fully qualified MVS dataset name'
: The complete name of the MVS data set that contains the OMPROUTE configuration statements. The data set can be a sequential data set or a PDS with the member name.

  Requirement: The double slash (//) and beginning and ending quotation marks are required.

/file system absolute pathname
: The complete name of the file system that contains the OMPROUTE configuration statements. The z/OS UNIX path name is case sensitive.

  Requirement: The beginning slash (/) is required.

  Guideline: If a syntax error is encountered in the final version of the configuration file after one or more INCLUDE files were processed, use debug level d1 to print a copy of the expanded configuration file to your OMPROUTE trace. This helps to identify the correct line number where the syntax error was found, as reported from the error message. For more information about OMPROUTE traces and debug information, see z/OS Communications Server: IP Diagnosis Guide.

OSPF configuration statements

This topic contains descriptions of the following OSPF configuration statements:
- AREA
- AS_BOUNDARY_ROUTING
- COMPARISON
- DEMAND_CIRCUIT
- OSPF
- OSPF_INTERFACE
- RANGE
- ROUTERID
- VIRTUAL _LINK

Use these statements to configure the OSPF environment for IPv4. For information about the statements used to configure IPv6 OSPF, see “IPv6 OSPF configuration statements” on page 518.

See z/OS Communications Server: IP System Administrator's Commands for information about how to display configuration information.

AREA statement

Use the AREA statement to set the parameters for an OSPF area. If no areas are defined, OMPROUTE assumes that all directly attached networks belong to the backbone area (area ID 0.0.0.0).

Syntax

```
            ┌─Area_Number=0.0.0.0────────────────────┐
▶▶──Area────┤                                        ├──────────────────────────────────▶
            └─Area_Number──=──ospf_area_address──────┘

  ┌─Authentication_Type=None──────────────────┐   ┌─Stub_Area=NO──────────┐
▶─┤                                           ├───┤                       ├──────────────▶
  └─Authentication_Type──=──security_scheme───┘   └─Stub_Area──=──value───┘

  ┌─Stub_Default_Cost=1──────────┐   ┌─Import_Summaries=YES──────────┐
▶─┤                              ├───┤                               ├──────────────────▶◀
  └─Stub_Default_Cost──=──cost───┘   └─Import_Summaries──=──value────┘
```

Parameters

Area_Number
The OSPF area number in dotted decimal.

Authentication_Type
The default security scheme to be used in the area. Valid values for authentication types are MD5, which indicates MD5 cryptographic authentication as described in Appendix D of RFC 2328; PASSWORD, which indicates a simple password; or NONE, which indicates that no authentication is necessary to pass packets. The area's default security scheme can be overridden on an interface basis by specifying the Authentication_Type keyword on OSPF_INTERFACE or VIRTUAL_LINK statements.

Stub_Area
Specifies whether this area is a stub area or not. Valid values are YES or NO.

If you specify Stub_area = YES, the area does not receive any AS external link advertisements, reducing the size of your database and decreasing memory usage for routers in the stub area. You cannot configure virtual links through a stub area. You cannot configure a router within the stub area as an AS boundary router.

You cannot configure the backbone as a stub area. External routing in stub areas is based on a default route. Each border area router attaching to a stub area originates a default route for this purpose. The cost of this default route is also configurable with the AREA statement.

Stub_Default_Cost
The cost that an OMPROUTE area border router associates with the default route that it generates into the stub area. Valid values are in the range 1 - 16 777 215.

Import_Summaries
Determines whether this stub area mports a routing summary from a neighbor area. Valid values are YES or NO.

AS_BOUNDARY_ROUTING statement

Use the AS_BOUNDARY_ROUTING statement to enables the AS boundary routing capability, which allows you to import routes learned from other methods (RIP, statically configured, and direct routes) into the OSPF domain. All routes are imported as either Type 1 or Type 2 external routes, depending on what was coded on the Comparison statement. The metric type used when importing routes determines how the imported cost is viewed by the OSPF domain. When comparing Type 2 metrics, only the external cost is considered in picking the best route. When comparing Type 1 metrics, the external and internal costs of the route are combined before making the comparison.

Rules:
- This statement must be coded even if the only route you want to import is the default route (destination 0.0.0.0).
- You can import into the OSPF domain only static routes that are configured in the main route table. You cannot import static routes that are configured in a policy-based route table.

Syntax

```
                          ┌─Import_RIP_Routes=No──────┐
►►──AS_Boundary_Routing───┼───────────────────────────┼──────────────────────────────►
                          └─Import_RIP_Routes──=──value┘

   ┌─Import_Static_Routes=No──────┐   ┌─Import_Direct_Routes=No──────┐
►──┼─────────────────────────────┼───┼─────────────────────────────┼────────────────►
   └─Import_Static_Routes──=──value┘   └─Import_Direct_Routes──=──value┘

   ┌─Import_Subnet_Routes=Yes─────┐   ┌─Originate_Default_Route=No──────┐
►──┼─────────────────────────────┼───┼────────────────────────────────┼─────────────►
   └─Import_Subnet_Routes──=──value┘   └─Originate_Default_Route──=──value┘

   ┌─Originate_as_Type=2─────────┐   ┌─Default_Route_Cost=1─────────┐
►──┼────────────────────────────┼───┼─────────────────────────────┼──────────────────►
   └─Originate_as_Type──=──type─┘   └─Default_Route_Cost──=──cost─┘

   ┌─Learn_Default_Route=NO──────┐
►──┼────────────────────────────┼────────────────────────────────────────────────────►
   └─Learn_Default_Route──=──value┘

►──┬──────────────────────────────────────────┬──────────────────────────────────────►◄
   └─Default_Forwarding_Address──=──address───┘
```

Parameters

Import_RIP_Routes
: Specifies whether routes learned by RIP are imported into the OSPF routing domain. Valid values are YES or NO.

Import_Static_Routes
: Specifies whether static routes (routes defined to the TCP/IP stack using the BEGINROUTES or GATEWAY statement) are imported into the OSPF routing domain. Valid values are YES or NO.

Import_Direct_Routes
: Specifies whether direct routes are imported into the OSPF routing domain. Valid values are YES or NO.

Import_Subnet_Routes
: Independent of the RIP, static, and direct routes you can choose to import, you can also configure whether or not to import subnet routes into the OSPF domain. Valid values are YES or NO.

Originate_Default_Route
: Specifies whether or not this router originates an AS External default route into

the OSPF domain. If YES and Default_Forwarding_Address is not also coded (or is coded to 0.0.0.0), this router advertises itself as a default router. Valid values are YES or NO.

`Originate_as_Type`
: Specifies the external type assigned to the default route. Valid values are 1 or 2.

`Default_Route_Cost`
: Specifies the cost that OSPF associates with the default route. Valid values are in the range 0 - 16 777 215.

`Learn_Default_Route`
: Specifies whether OSPF learns default routes from inbound RIP or OSPF external packets when their cost is equal to or higher than the default route originated by this host. Valid values are YES or NO.

`Default_Forwarding_Address`
: If Originate_Default_Route is YES, this optional parameter can be used to specify that this router should originate a default route on behalf of a different router. This parameter is not needed if this router is to advertise itself as the default router. It should be used only when the default router is another router that this router can route to, which is not capable of advertising an OSPF default route on its own behalf. In that case, this parameter should be set to a reachable interface IP address on the other router.

  Restriction: This address must be reachable using an OSPF intra-area or inter-area route (labelled as SPF or SPIA in the RTTABLE display, or labelled as DIR but using an OSPF interface). This route could be a host, subnet, network, or default route. If no eligible route is found, the forwarding address is not included in the advertisements generated by this statement.

COMPARISON statement

Use the COMPARISON statement as an alternate method for specifying the Comparison parameter on the OSPF configuration statement. See “OSPF statement” on page 492 for a description of this statement.

For additional information about the COMPARISON configuration statement, see z/OS Communications Server: IP Configuration Guide.

Syntax

```
   ┌─Comparison=Type2───────┐
►►─┴─Comparison──=──value───┴──────────────────────────────────────────►◄
```

Parameters

`Comparison`
: Compare to type 1 or 2 externals. Valid values are Type1 (or 1) or Type2 (or 2).

DEMAND_CIRCUIT statement

Use the DEMAND_CIRCUIT statement as an alternate method for specifying the DEMAND_CIRCUIT parameter on the OSPF configuration statement. See “OSPF statement” on page 492 for a description of this statement.

Syntax

```
   ┌─Demand_Circuit=YES─────┐
►►─┼────────────────────────┼──────────────────────────────────────────────►◄
   └─Demand_Circuit──=──value┘
```

Parameters

Demand_Circuit
Valid values are YES or NO.

OSPF statement

Use the OSPF statement to specify parameters that apply globally to IPv4 OSPF, either to all interfaces or to the overall OSPF autonomous system.

The following parameters can also be specified as stand-alone configuration statements:

- DEMAND_CIRCUIT
- ROUTERID
- COMPARISON

Guideline: You should use the OSPF statement for defining these parameters. If both the OSPF statement and the standalone statements are coded, the last one coded in the configuration file takes precedence.

Syntax

```
                                 ┌─Comparison──=──Type2─┐
►►──OSPF──┬─────────────────────┬┼──────────────────────┼──────────────────────►
          └─RouterID──=──value──┘└─Comparison──=──value─┘

 ┌─Demand_Circuit──=──YES───┐ ┌─DR_Max_Adj_Attempt──=──0─────┐
►┼──────────────────────────┼─┼──────────────────────────────┼──────────────►◄
 └─Demand_Circuit──=──value─┘ └─DR_Max_Adj_Attempt──=──value─┘
```

Parameters

RouterID
Every router in an IPv4 OSPF routing domain must be assigned a unique 32-bit router ID.

The value used for the OSPF router ID is chosen as follows:

- If this RouterID statement is specified, the value configured is used as the OSPF router ID. This value must be one of the OSPF interface IP addresses that is configured for the stack.

  Rule: Loopback and reserved 0.0.0.0 addresses are not valid IP interface addresses.
- If the router ID is not configured, one of the OSPF interface addresses will be used as the OSPF router ID, provided that the interface exists in the TCPIP profile and matches the corresponding OSPF interface statement in the OMPROUTE configuration file at startup.

  Result: When OMPROUTE has to assign the router ID, it does not use dynamic VIPA IP addresses. This avoidance of dynamic VIPA IP addresses cannot be guaranteed; for example, if dynamic VIPAs are the only active OSPF interfaces when OMPROUTE chooses the router ID, then one of them will be chosen.

Guideline: Because dynamic VIPAs (DVIPAs) can move between z/OS hosts, the router ID should be a physical interface or a static VIPA, not a dynamic VIPA address. To ensure an appropriate router ID, specify the router ID to OMPROUTE.

Valid values are any IPv4 dotted-decimal address that matches a configured OSPF interface.

Comparison
Tells OMPROUTE where external routes fit in the IPv4 OSPF hierarchy. OSPF supports two types of external metrics. Type 1 external metrics are equivalent to the link state metric. Type 2 external metrics are greater than the cost of any path internal to the autonomous system. Use of type 2 external metrics assumes that routing between autonomous systems is the major cost of routing a packet, and eliminates the need for conversion of external costs to internal link state metrics. For more information about the COMPARISON configuration parameter, see z/OS Communications Server: IP Configuration Guide. Valid values are Type1 (or 1) or Type2 (or 2).

Demand_Circuit
This value determines the global demand circuit setting. Coding YES enables demand circuits. Demand circuit parameters can then be coded on the OSPF_Interface statement. Valid values are Yes or No.

DR_Max_Adj_Attempt
Specifies the maximum number of adjacency attempts to be used for reporting and controlling futile neighbor state loops. After the adjacency attempt count for a neighboring designated router reaches the threshold, an informational message is issued to report the problem. If a redundant interface is available that can reach the neighbor, adjacency formation is attempted over that interface. An informational message is issued to report the interface switch and adjacency formation attempt. Valid values are in the range 0 - 100. The value 0 specifies infinite retries.

For information about futile neighbor state loops, see the network design considerations information in z/OS Communications Server: IP Configuration Guide. For the types of interfaces that support the futile neighbor state loop detection for OSPF, see "Interfaces supported by OMPROUTE" on page 548.

OSPF_INTERFACE statement

Use the OSPF_INTERFACE statement to set the OSPF parameters for interfaces. Replicate this statement in the configuration file for each IP interface over which OSPF operates.

Syntax

```
►►──OSPF_Interface──IP_address──=──ip_address──Name──=──interface_name────────────►
```

```
►─Subnet_Mask──=──subnet_mask─┬──────────────────────────────┬──────────────►
                              └─Destination_Addr──=──address─┘


  ┌─Attaches_To_Area=0.0.0.0─┐  ┌─MTU=576──────────┐
►─┼──────────────────────────┼──┼──────────────────┼──────────────────────────►
  └─Attaches_To_Area──=──area┘  └─MTU──=──mtu_size─┘

  ┌─Retransmission_Interval=5─────────────┐  ┌─Transmission_Delay=1──────────┐
►─┼───────────────────────────────────────┼──┼───────────────────────────────┼─►
  └─Retransmission_Interval──=──frequency─┘  └─Transmission_Delay──=──delay──┘

  ┌─Router_Priority=1─────────────┐  ┌─Hello_Interval=10──────────┐
►─┼───────────────────────────────┼──┼────────────────────────────┼───────────►
  └─Router_Priority──=──priority──┘  └─Hello_Interval──=──interval┘

  ┌─DB_Exchange_Interval=40──────────┐  ┌─Dead_Router_Interval=40──────────┐
►─┼──────────────────────────────────┼──┼──────────────────────────────────┼──►
  └─DB_Exchange_Interval──=──interval┘  └─Dead_Router_Interval──=──interval┘

  ┌─Cost0=1──────────┐  ┌─Subnet=NO─────────┐
►─┼──────────────────┼──┼───────────────────┼─────────────────────────────────►
  └─Cost0──=──cost───┘  └─Subnet──=──value──┘

  ┌─Advertise_VIPA_Routes=Host_And_Subnet─┐
►─┼───────────────────────────────────────┼──┬──────────────────────────┬─────►
  └─Advertise_VIPA_Routes──=──value───────┘  └─Authentication_type=value┘

  ┌─Authentication_Key_ID=0─┐  ┌─Authentication_Key=nulls─────────┐
►─┼─────────────────────────┼──┼──────────────────────────────────┼───────────►
  └─Authentication_Key_ID=id┘  └─Authentication_Key──=──password──┘

  ┌─Demand_Circuit=no────────┐  ┌─Hello_Suppression=Allow──────┐
►─┼──────────────────────────┼──┼──────────────────────────────┼──────────────►
  └─Demand_Circuit──=──value─┘  └─Hello_Suppression──=──value──┘

  ┌─PP_Poll_Interval=60──────────┐  ┌─Parallel_OSPF=Backup──────┐
►─┼──────────────────────────────┼──┼───────────────────────────┼─────────────►
  └─PP_Poll_Interval──=──interval┘  └─Parallel_OSPF──=──value───┘

  ┌─Non_Broadcast=NO─────────┐  ┌─NB_Poll_Interval=120─────────┐
►─┼──────────────────────────┼──┼──────────────────────────────┼──────────────►
  └─Non_Broadcast──=──value──┘  └─NB_Poll_Interval──=──interval┘

  ┌────────────────────────────┐  ┌────────────────────────────────┐
  ▼                            │  ▼                                │
►───┬────────────────────────┬─┴────┬────────────────────────────┬─┴──────────►
    └─DR_Neighbor──=──value──┘      └─No_DR_Neighbor──=──value───┘
```

```
   ┌─Max_Xmit_Time=120──────┐  ┌─Min_Xmit_Time=0.5──────┐
►──┼────────────────────────┼──┼────────────────────────┼────────────────────────►
   └─Max_Xmit_Time──=──time─┘  └─Min_Xmit_Time──=──time─┘

   ┌─RT_Gain=0.125────────┐ ┌─Variance_Gain=0.25─────────┐
►──┼──────────────────────┼─┼────────────────────────────┼───────────────────────►
   └─RT_Gain──=──value────┘ └─Variance_Gain──=──value────┘

   ┌─Variance_Mult=2────────┐  ┌─Delay_Acks=YES─────────┐
►──┼────────────────────────┼──┼────────────────────────┼───────────────────────►◄
   └─Variance_Mult──=──mult─┘  └─Delay_Acks──=──value───┘
```

Parameters

IP_address

IP address of the local interface to be configured for OSPF.

The IP address can be a valid IP address that is configured on the system, or it can be specified with asterisks (*) as wildcards. The valid wildcard specifications are shown in the following example. The result of coding a wildcard value is that all configured interfaces whose IP address matches the wildcard are configured as OSPF interfaces. Configured interface IP addresses and names are matched against possible wildcards in the order they appear in the following example with the name and any matching wildcard being the best match, x.y.z.* being second best, and so on.

```
interface name and any matching wildcard
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.* - Same as ALL
ALL - Same as *.*.*.*
```

Tip: For more information about how wildcard interfaces are parsed, see this Method of assigning interface definitions to stack interfaces (wildcard and explicit): in z/OS Communications Server: IP Configuration Guide.

Because a stack could have a large number of Dynamic VIPAs (DVIPAs) defined, as well as DVIPA ranges, an additional wildcard capability exists on the OSPF_INTERFACE statement for use only with DVIPAs. Ranges of DVIPA interfaces can be defined using the subnet mask parameter on the OSPF_INTERFACE statement. This mode of definition applies to Dynamic VIPAs defined in the stack with VIPADEFINE, VIPABACKUP, or VIPARANGE. The range defined in this way are all the IP addresses that fall within the subnet defined by the mask and the IP address. When this type of wildcarding is being used, the value of the IP_ADDRESS parameter must be the subnet number of the range. Note that this subnet number is not equivalent to a DVIPA address as defined in VIPADEFINE or VIPABACKUP parameter of the VIPADYNAMIC statement in the TCP/IP profile. If VIPARANGE is defined, you should code DVIPA subnet address for the subnet number. For example, the following code defines a range of six addresses (9.67.101.9 to 9.67.101.14) that can be used for DVIPA addresses and matches any DVIPA interface that falls into the 9.67.101.8/29 subnet:

```
IP_ADDRESS = 9.67.101.8
SUBNET_MASK = 255.255.255.248
```

Alternatively, the following code does not because 9.67.101.17 is an address within the subnet range, not the subnet number itself (that would be 9.67.101.16). This second definition matches only an interface whose home address is 9.67.101.17.

```
IP_ADDRESS= 9.67.101.17
SUBNET_MASK=255.255.255.248
```

Name
The name of the interface. A valid value is any string 1 - 16 characters in length.

Rules:

- If this is not a wildcard interface definition, the name must match the link name that is coded for the corresponding IP address on the HOME statement or the interface name coded for the corresponding IPv4 INTERFACE statement in the TCP/IP profile.
- If this is a wildcard interface definition, then this parameter is used in conjunction with the defined wildcard IP address when searching for definitions to match a stack interface. For more details about this process, see method of assigning interface definitions to stack interfaces (wildcard and explicit): in z/OS Communications Server: IP Configuration Guide.

Subnet_Mask
The subnet mask of the subnet to which this interface attaches. This value must be the same for all routers attached to a common network. For more information, see z/OS Communications Server: IP Configuration Guide. If you configure this interface in the TCP/IP profile using the IPv4 INTERFACE statement and you configure a subnet mask on that statement that does not match the value that you specify on this parameter, OMPROUTE issues message EZZ8164I and uses this subnet mask.

Destination_Addr
IP address of the host at the remote end of this interface. This parameter is valid only for point-to-point links. If this parameter is not specified for a point-to-point link, a route to the host at the remote end of the interface is not added to the appropriate TCP/IP route tables (main and policy-based tables) until OSPF communication is established with that host. A subnet route for the interface is added when OMPROUTE is initialized whether or not this parameter is specified.

Attaches_To_Area
OSPF area to which this interface attaches. Valid values are 0.0.0.0 (the backbone), or any area defined by the AREA statement.

MTU
The maximum transmission unit size that OSPF adds to the appropriate routing tables (main and policy-based tables) for routes that use this interface. Valid values are in the range 0 - 65535. If you configure this interface in the TCP/IP profile using the IPv4 INTERFACE statement and you configure an MTU on that statement and the MTU that you configure on that statement does not match the MTU (the configured value or the default value) on this statement, OMPROUTE issues message EZZ8163I and uses the MTU value on this statement.

Tip: See z/OS Communications Server: IP Configuration Guide, in section Maximum transmission unit considerations, for additional information about how TCP/IP uses the MTU to determine the largest size frame to send.

Retransmission_Interval
Sets the frequency (in seconds) of retransmitting link-state update packets, link-state request packets, and database description packets. Valid values are in the range 1 - 65535 seconds.

If this parameter is set too low, needless retransmissions occur that could affect performance and interfere with neighbor adjacency establishment. It should be set to a higher value for a slower machine.

Transmission_Delay
This parameter is an estimate of the number of seconds that it takes to transmit link-state information over the interface. Each link-state advertisement has a finite lifetime of 1 hour. As each link-state advertisement is sent out from this interface, it is aged by this configured transmission delay. Valid values are in the range 1 - 65535 seconds.

Router_Priority
This value is used for broadcast and nonbroadcast multiaccess networks to elect the designated router, with the highest priority router being elected. Valid values are in the range 0 - 255.

A value of 0 indicates that OMPROUTE never becomes the designated router. A value of 1 indicates the lowest possible eligible priority and a value of 255 indicates the highest possible priority.

Hello_Interval
This parameter defines the number of seconds between OSPF Hello packets being sent out on this interface. This value must be the same for all routers attached to a common network. Valid values are in the range 1 - 255 seconds.

DB_Exchange_Interval
The interval in seconds that the database exchange process cannot exceed. If the interval elapses, the procedure is restarted. This value must be larger than the Hello_Interval. If no value is specified, the DB_Exchange_Interval is set to the Dead_Router_Interval. Valid values are 2 through 65535.

Dead_Router_Interval
The interval in seconds, after not having received an OSPF Hello, that the neighbor is declared to be down. This value must be larger than the Hello_Interval. Setting this value too close to the Hello_Interval can result in the collapse of adjacencies. A value of 4*Hello_Interval is preferred. This value must be the same for all routers attached to a common network. Valid values are 2 - 65535.

Cost0
The OSPF cost for this interface. The cost is used to determine the shortest path to a destination. Valid values are in the range 1 - 65535.

Subnet
The meaning of this parameter depends on the interface type.

For an interface to a point-to-point link, this option enables the advertisement of a stub route to the subnet that represents the link rather than the host route for the other router's address. In effect, this parameter controls whether, for this interface, OMPROUTE implements option 1 (SUBNET=NO) or option 2 (SUBNET=YES) described in RFC 2328 (OSPF version 2) topic 12.4.1.1. For a detailed explanation of this option, see the IPv4 interface information in z/OS Communications Server: IP Configuration Guide.

For a VIPA interface, this option suppresses advertisement of either the VIPA host or subnet route. Normally z/OS Communications Server advertises both a host route and a subnet route for owned VIPA interfaces. With this option set to NOVIPAHOST, the VIPA host route is suppressed and only the VIPA subnet route is advertised. With this option set to NOVIPASUBNET, the VIPA subnet route is suppressed and only the VIPA host route is advertised.

Legal values are:
- YES
- NO
- NOVIPASUBNET
- NOVIPAHOST

Guidelines:
- Using the NOVIPAHOST value has the same effect as setting SUBNET=YES or ADVERTISE_VIPA_ROUTES=SUBNET_ONLY, using the NOVIPASUBNET value is equivalent to setting ADVERTISE_VIPA_ROUTES=HOST_ONLY
- The ADVERTISE_VIPA_ROUTES option is the preferred method to suppress VIPA advertisements.

Rule: Do not use this option for dynamic VIPAs or for any VIPA whose subnet might exist on multiple hosts. If you do, problems can occur routing to all VIPAs that share the subnet.

Tips:
- Specifying SUBNET=YES on a VIPA interface has the same effect as specifying SUBNET=NOVIPAHOST.
- In order to fully suppress the VIPA subnet route, SUBNET=NOVIPASUBNET must be specified on every VIPA OSPF_INTERFACE statement that defines a VIPA in a common subnet.

`Advertise_VIPA_Routes`

This option is valid only on VIPA interfaces and controls how OMPROUTE advertises the VIPA address. The default value of HOST_AND_SUBNET advertises both the VIPA host and subnet route. With this option set to HOST_ONLY, only the VIPA host route is advertised. With this option set to SUBNET_ONLY, only the VIPA subnet route is advertised.

The value specified on the ADVERTISE_VIPA_ROUTES option overrides any value specified on the SUBNET option. Legal values are:
- HOST_AND_SUBNET
- HOST_ONLY
- SUBNET_ONLY

Rule: Do not specify SUBNET_ONLY for dynamic VIPAs or for any VIPA whose subnet might exist on multiple hosts. Problems can occur routing to all VIPAs that share the subnet when the subnet exists on multiple hosts.

Tip: The HOST_ONLY option must be specified for every VIPA in a common subnet. If the HOST_ONLY option is not specified for every VIPA in a common subnet, OMPROUTE still advertises the VIPA subnet route for the interfaces not specifying HOST_ONLY.

`Authentication_Type`

The security scheme to be used on the network to which the interface attaches. If parameter is not specified, takes on the default value specified for the area to which the interface is attached. Valid values for authentication types are MD5, which indicates MD5 cryptographic authentication as described in Appendix D of RFC 2328; PASSWORD, which indicates a simple password; or NONE, which indicates that no authentication is necessary to pass packets. All hosts on the network must be configured with the same security scheme.

Authentication_Key_ID
The identifier of the authentication key defined with the AUTHENTICATION_KEY keyword. This is a constant numeric value from 0 - 255, with a default value of 0. It is relevant only when MD5 cryptographic authentication is employed on the interface; otherwise, it is ignored. This field is provided for compatibility with other routers that might require identification of a key identifier with the authentication key.

Authentication_Key
The value of the authentication key for this interface. This value must be the same for all routers attached to a common medium. The coding of this parameter depends on the authentication type being used on this interface.

For authentication type *none*, this parameter is not required and is ignored if coded.

For authentication type *password*, code the password for OSPF routers that are attached to this subnet. Valid values are any characters from EBCDIC code page 1047 up to 8 characters in length coded within double quotation marks or any hexadecimal string up to 8 bytes (16 hexadecimal characters) long that begins with 0x.

For authentication type *MD5*, code the 16-byte MD5 authentication key for OSPF routers attached to this subnet. This value can be coded in one of the following ways:

- The standard method is with a 16-byte hexadecimal string beginning with 0x (0x plus 32 hexadecimal characters). In some cases, pwtokey can be used to generate hexadecimal MD5 keys. See z/OS Communications Server: IP System Administrator's Commands for more information.
- An additional method, which provides compatibility with Cisco, Extreme, and other vendor routers that use a Cisco-compatible CLI interface is to code the MD5 key as an ASCII string, specified in double quotation marks prefixed with A. For example, to be compatible with this Cisco key definition, use the following code:

  ```
  ip ospf message-digest-key 4 md5 ABCDEFGHIJKLMNOP
  ```

  This value would be coded in OMPROUTE as follows:

  ```
  AUTHENTICATION_KEY_ID =4
  AUTHENTICATION_KEY = A"ABCDEFGHIJKLMNOP"
  ```

Demand_Circuit
This parameter, when coded with YES, causes Link State Advertisements (LSAs) to not be periodically refreshed over this interface. Only LSAs with real changes are advertised. In addition, coding this parameter to YES causes LSAs flooded over this interface to never age out. Valid values are YES or NO. For more information about the Demand_Circuit=YES and related topics, such as handling high cost links, see z/OS Communications Server: IP Configuration Guide.

Hello_Suppression
This parameter is used only on point-to-point and point-to-multipoint interfaces that are demand circuits. It allows you to configure the interface for Hello Suppression. Valid values are ALLOW, REQUEST, or DISABLE.

If either or both sides specify DISABLE, Hello_Suppression is disabled. If both specify ALLOW, Hello_Suppression is disabled. If one specifies ALLOW and the other REQUEST, or if both specify REQUEST, Hello_Suppression is enabled.

PP_Poll_Interval
This parameter specifies the interval (in seconds) that OMPROUTE should use when attempting to contact a neighbor to reestablish a neighbor relationship when the relationship has failed, but the interface is still available. This parameter is meaningful only if Demand_Circuit is coded YES and Hello_Supression has been enabled. Valid values are in the range 0 - 65535.

Parallel_OSPF
This parameter designates whether the OSPF interface is primary or backup when more than one OSPF interface is defined to the same subnet. Only one of these interfaces can be configured as primary, meaning that it is the interface to carry the OSPF protocol traffic between OMPROUTE and the subnet. Failure of the primary interface results in automatic switching of OSPF traffic to one of the backup interfaces. If the primary interface is later reactivated, OSPF traffic is not automatically switched back from the backup interface to the primary interface. If you want to switch OSPF traffic back to the primary interface, stop the backup interface. If none of the interfaces to the common subnet are configured as primary, a primary interface is selected by OMPROUTE. Valid values are BACKUP and PRIMARY.

Non_Broadcast
If the router is connected to a nonbroadcast, multiaccess network (NBMA), such as X.25, Frame Relay, Hyperchannel, or ATM networks, coding a Non_Broadcast helps the router discover its neighbors. This can also be coded for a broadcast-capable network when you want OMPROUTE to unicast its packets instead of multicasting them. In addition to coding this parameter, each neighbor must be configured with the DR_NEIGHBOR parameter, for those neighbors that are eligible to become the designated router, or NO_DR_NEIGHBOR for those neighbors that are not eligible to become the designated router. This statement is ignored when this OSPF interface is coded as a wildcard. Valid values are YES or NO.

NB_Poll_Interval
This parameter specifies the frequency (in seconds) of hellos sent to neighbors that are inactive. You must set this poll interval consistently across all interfaces that attach to the same subnetwork for OSPF to function correctly. This statement is valid only when Non_Broadcast is coded as YES. Valid values are in the range 1 - 65535.

DR_Neighbor
Configures the IP interface address of a designated router-eligible neighbor adjacent to the router over this interface. In nonbroadcast multi-access networks, neighbors need to be configured to all OSPF routers on the network. Multiple DR_Neighbor statements can be coded on an OSPF_interface statement as necessary.

Guideline: You should not define neighbors on broadcast-capable or multicast-capable media. If you do define neighbors on these media, OMPROUTE can communicate OSPF information only with those neighbors that are defined (it does not form adjacencies with any additional neighbors).

No_DR_Neighbor
Configures the IP interface address of a nondesignated router-eligible neighbor adjacent to the router over this interface. In nonbroadcast multi-access

networks, neighbors need to be configured to all OSPF routers on the network. Multiple No_DR_Neighbor statements can be coded on an OSPF_Interface statement as necessary.

Guideline: You should not define neighbors on broadcast-capable or multicast-capable media. If you do define neighbors on these media, OMPROUTE can communicate OSPF information only with those neighbors that are defined (it does not form adjacencies with any additional neighbors).

Retransmit Parameters

The following parameters are used by OMPROUTE to set values in the routes that use this interface; the values are added to the TCP/IP route tables. The values affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a certain number of retransmissions, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times packets are retransmitted before the connection times out differs for initial connection establishment and for data packets. For initial connection establishment, the initial time interval is set at approximately 3 seconds and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval and are retransmitted 15 times before the connection is timed out. All of the following parameters affect the data packet retransmission algorithm. Only the Min_Xmit_Time parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

`Max_Xmit_Time`
Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to time out. Specifying Max_Xmit_Time assures that the interval time never exceeds the specified limit. The minimum value that can be specified for Max_Xmit_Time is 0. The maximum is 999.990. The default is 120 seconds. This parameter affects the initial connection establishment retransmission timeout for all APIs, except the Pascal API (TcpOpen), that are using the socket connect function.

`Min_Xmit_Time`
Sets a minimum retransmit interval. Increasing this value might increase the amount of time it takes for TCP to time out a connection. The minimum value that can be specified for Min_Xmit_Time is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

`RT_Gain`
This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet's RTT has on the average. The minimum value that can be specified for RT_Gain is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

Variance_Gain
This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum value that can be specified for Variance_Gain is 0. The maximum value is 1.0. The default is 0.25 . This parameter does not affect initial connection retransmission.

Variance_Mult
This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT has on calculating the retransmission interval. The minimum value that can be specified for Variance_Mult is 0. The maximum value is 99.990. The default is 2. This parameter does not affect initial connection retransmission.

Delay_Acks
The delay acknowledgments value that is added to the routing tables for routes that use this interface. Specify YES to delay transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specify NO to return acknowledgments immediately when a packet is received with the PUSH bit on in the TCP header. This parameter affects only connections that use the routes associated with this interface.

Even if you specify YES, you can override the delay acknowledgments behavior can be overridden by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements. The value NO can override the specification the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

Valid values are YES and NO. The default value is YES.

Usage notes

When you configure multiaccess parallel interfaces (primary and secondary interfaces that have IP addresses in the same subnet) for OMPROUTE (OSPF), code the Parallel_OSPF=Primary parameter to set a specific interface as the primary interface. If none of the interfaces on the same subnet are coded as primary, OMPROUTE will select the primary interface from the set of interfaces attached to the subnet. In case of a primary interface failure, OMPROUTE uses the first available secondary interface and marks it as the primary interface.

RANGE statement

Use the RANGE statement to add ranges to OSPF areas. OSPF areas can be defined in terms of address ranges. External to the area, a single route is advertised for each address range. For example, if an OSPF area were to consist of all subnets of the class B network 128.185.0.0, it would be defined as consisting of a single address range. The address range would be specified as an address of 128.185.0.0 together with a mask of 255.255.0.0. Outside of the area, the entire subnetted network would be advertised as a single route to network 128.185.0.0.

Ranges can be defined to control which routes are advertised external to an area.

When OSPF is configured not to advertise the range, no interarea routes are advertised for routes that fall within the range. Ranges cannot be used for areas that serve as transit areas for virtual links. This does not prevent AS-external routes from being advertised if used in conjunction with the AS_BOUNDARY statement.

Syntax

```
►►──Range──IP_address──=──address──Subnet_Mask──=──mask──────────────────────────►

   ┌─Area_Number=0.0.0.0──┐   ┌─Advertise=YES─────────┐
►──┼──────────────────────┼───┼───────────────────────┼──────────────────────────►◄
   └─Area_Number──=──area─┘   └─Advertise──=──value───┘
```

Parameters

IP_Address
Common subnet portion of IP addresses in this range. Valid values are valid network and subnetwork addresses.

Subnet_Mask
Subnet mask with respect to the network range defined in IP_Address.

Area_Number
Area number for which to add this range. Valid values are any defined areas.

Advertise
Determines whether this range is advertised to other areas. Valid values are YES or NO.

RouterID statement

Use the RouterID statement as an alternative to specifying the RouterID parameter on the OSPF configuration statement. See "OSPF statement" on page 492 for the statement descriptions. The following concepts apply to the RouterID statement:

- If the RouterID statement is specified, the value configured is used as the OSPF router ID. This value must be one of the OSPF interface IP addresses that is configured for the stack.

  Rule: Loopback and reserved 0.0.0.0 addresses are not valid OSPF interface IP addresses.
- If the router ID is not configured, one of the OSPF interface addresses will be used as the OSPF router ID, provided that the interface exists in the TCPIP profile and matches the corresponding OSPF interface statement in the OMPROUTE configuration file at startup.

  Result: When OMPROUTE has to assign the router ID, it does not use dynamic VIPA IP addresses. This avoidance of dynamic VIPA IP addresses cannot be guaranteed; for example, if dynamic VIPAs are the only active OSPF interfaces when OMPROUTE chooses the router ID, then one of them will be chosen.

  Guideline: Because dynamic VIPAs (DVIPAs) can move between z/OS hosts, the router ID should be a physical interface or a static VIPA, not a dynamic VIPA address. To ensure an appropriate router ID, specify the router ID to OMPROUTE.

Syntax

```
►►──RouterID──=──id──────────────────────────────────────────────────────────────►◄
```

Parameters

RouterID
A dotted-decimal value.

VIRTUAL_LINK statement

Use the VIRTUAL_LINK statement to configure a virtual link between two area border routers. To maintain backbone connectivity you must have all of your backbone routers interconnected either by permanent or virtual links. Virtual links are considered to be separate router interfaces connecting to the backbone area. Therefore, you are asked to specify many of the interface parameters when configuring a virtual link.

Virtual links can be configured between any two backbone routers that have an interface to a common nonbackbone, nonstub area. Virtual links are used to maintain backbone connectivity and must be configured at both endpoints.

Tip: OSPF virtual links are not to be confused with Virtual IP Address support (VIPA).

Syntax

```
►►──Virtual_Link──Virtual_Endpoint_RouterID──=──id─────────────────────────────►


   ┌─Links_Transit_Area=0.0.0.1──┐  ┌─Retransmission_Interval=10─────────────┐
►──┼─────────────────────────────┼──┼────────────────────────────────────────┼──►
   └─Links_Transit_Area──=──area─┘  └─Retransmission_Interval──=──frequency──┘

   ┌─Transmission_Delay=5─────────┐  ┌─Hello_Interval=30───────────┐
►──┼──────────────────────────────┼──┼─────────────────────────────┼────────────►
   └─Transmission_Delay──=──delay─┘  └─Hello_Interval──=──interval─┘

   ┌─DB_Exchange_Interval=180───────────┐  ┌─Dead_Router_Interval=180───────────┐
►──┼────────────────────────────────────┼──┼────────────────────────────────────┼──►
   └─DB_Exchange_Interval──=──interval──┘  └─Dead_Router_Interval──=──interval──┘

   ┌─Authentication_Key=nulls──────────┐  ┌─Authentication_Key_ID=0──┐
►──┼───────────────────────────────────┼──┼──────────────────────────┼──────────►
   └─Authentication_Key──=──password───┘  └─Authentication_Key_ID=id─┘

►──┬─────────────────────────────┬─────────────────────────────────────────────►◄
   └─Authentication_type=value───┘
```

Parameters

Virtual_Endpoint_RouterID
Router ID of the virtual neighbor (other endpoint). Router IDs are entered in the same form as IP addresses.

Links_Transit_Area
This is the nonbackbone, nonstub area through which the virtual link is configured. Virtual links can be configured between any two area border routers that have an interface to a common nonbackbone and nonstub area. Virtual links must be configured in each of the link's two endpoints. Valid values are any area defined by the AREA statement, except 0.0.0.0.

Retransmission_Interval
Sets the frequency (in seconds) of retransmitting link-state update packets, link-state request packets, and database description packets. Valid values are from 1 - 65 535 seconds.

Guideline: If this parameter is set too low, needless retransmissions occur that could affect performance and interfere with neighbor adjacency establishment. It should be set to a higher value for a slower machine.

`Transmission_Delay`
This parameter is an estimate of the number of seconds that it takes to transmit link-state information over the virtual link. Each link-state advertisement has a finite lifetime of 1 hour. As each link-state advertisement is sent out from this virtual link, it is aged by this configured transmission delay. Valid values are in the range 1 - 65 535 seconds.

`Hello_Interval`
This parameter defines the number of seconds between OSPF Hello packets being sent out from this virtual link. Valid values are in the range 1 - 255 seconds. The Hello_Interval should be set higher than the same value used on the intervening, actual OSPF interfaces.

`DB_Exchange_Interval`
The interval in seconds that the database exchange process cannot exceed. If the interval elapses, the procedure is restarted. This value must be larger than the Hello_Interval. If no value is specified, the DB_Exchange_Interval is set to the Dead_Router_Interval. Valid values are 2 - 65 535.

`Dead_Router_Interval`
The interval in seconds, after not having received an OSPF Hello, that the neighbor is declared to be down. This value must be larger than the Hello_Interval. Valid values are 2 - 65 535. The dead router interval should be set higher than the same value used on the intervening, actual, OSPF interfaces.

`Authentication_Key`
The value of the authentication key for this interface. This value must be the same for all routers attached to a common medium. The coding of this parameter depends on the authentication type being used on this interface.

For authentication type *none*, this parameter is not required and is ignored if coded.

For authentication type *password*, code the password for OSPF routers that are attached to this subnet. Valid values are any characters from EBCDIC code page 1047 up to 8 characters in length coded within double quotation marks or any hexadecimal string up to 8 bytes (16 hex characters) long that begins with 0x.

For authentication type *MD5*, code the 16-byte MD5 authentication key for OSPF routers attached to this subnet. This value can be coded in one of the following ways:

- The standard method is with a 16-byte hexadecimal string beginning with 0x (0x plus 32 hexadecimal characters). In some cases, pwtokey can be used to generate hexadecimal MD5 keys. See z/OS Communications Server: IP System Administrator's Commands for more information.
- An additional method, which provides compatibility with Cisco, Extreme, and other vendor routers that use a Cisco-compatible CLI interface is to code the MD5 key as an ASCII string, specified in double quotation marks prefixed with A. For example, to be compatible with this Cisco key definition, use the following code:

  ```
  ip ospf message-digest-key 4 md5 ABCDEFGHIJKLMNOP
  ```

  This value would be coded in OMPROUTE as follows:

```
AUTHENTICATION_KEY_ID =4
AUTHENTICATION_KEY = A"ABCDEFGHIJKLMNOP"
```

`Authentication_Key_ID`
The identifier of the authentication key defined with the AUTHENTICATION_KEY keyword. This is a constant numeric value from 0 - 255, with a default value of 0. It is only relevant when MD5 cryptographic authentication is employed on the virtual link; otherwise, it is ignored. This field is provided for compatibility with other routers which might require identification of a key identifier with the authentication key.

`Authentication_Type`
The security scheme to be used over the virtual link. If not specified, the statement takes on the default value specified for the backbone area. Valid values for authentication types are MD5, which indicates MD5 cryptographic authentication as described in Appendix D of RFC 2328; PASSWORD, which indicates a simple password; or NONE, which indicates that no authentication is necessary to pass packets. Both hosts attached to the virtual link must be configured with the same security scheme.

RIP configuration statements

This topic contains descriptions of the following RIP configuration statements:
- ACCEPT_RIP_ROUTE
- FILTER
- IGNORE_RIP_NEIGHBOR
- ORIGINATE_RIP_DEFAULT
- RIP_INTERFACE
- SEND_ONLY

These statements are for configuring the RIP environment for IPv4. For information about the statements to be used for configuring IPv6 RIP, see “IPv6 RIP configuration statements” on page 529.

ACCEPT_RIP_ROUTE statement

Use the ACCEPT_RIP_ROUTE statement to allow a network, subnet, or host route to be accepted independent of whether the interface it was received on has the corresponding reception parameter enabled (network, subnet, or host). Routes added in this manner can be thought of as a list of exception conditions.

Restriction: Coding this statement does not enable updates for this destination to be received on RIP interfaces with RECEIVE_RIP=NO coded. Also, this does not override RIP version filters coded using the RECEIVE_RIP parameter on RIP_INTERFACE statements. For example, on a RIP_INTERFACE with RECEIVE_RIP=RIP2, a RIPV1 route that would otherwise be allowed by this statement is not received.

Syntax

```
►►──Accept_RIP_Route──IP_address──=──address──────────────────────────────────►◄
```

Parameters

`IP_address`
Destination route to be unconditionally accepted.

FILTER statement

Use the FILTER statement can be coded stand-alone in the OMPROUTE configuration file (nosend and noreceive only) to apply to all configured RIP interfaces.

Syntax

```
►►──filter──=──(filter_type,dest_route,filter_mask)──────────────────────────►◄
```

Parameters

filter_type
: The *filter_type* can be any of the following values:

  nosend
  : Specifies that routes matching the dest_route and filter_mask are not to be broadcast over RIP interfaces. This option serves as an RIP output filter.

  noreceive
  : Specifies that routes matching the dest_route and filter_mask are to be ignored in broadcasts received over RIP interfaces. This option serves as a RIP input filter.

dest_route
: The *dest_route* specifies the destination route in network, subnetwork, or host format in dotted decimal form. Alternatively, an asterisk (*), which matches *any* destination, can be coded to filter out all routes sent or received over an interface. The use of the asterisk is also referred to as a blackhole filter. This should be used in conjunction with either additional send or receive filters to allow only certain routes to be received, or advertised over an interface or set of interfaces.

  Tip: When the Originate_RIP_Default statement is configured, the blackhole nosend filter does not prevent sending of the default route.

filter_mask
: The *filter_mask* specifies the filter mask in dotted decimal form. If this value is not coded, the default filter mask is 255.255.255.255, meaning apply the filter to the *dest_route* as coded. Coding the filter mask has no meaning and is not valid if the *dest_route* is coded as an asterisk (*) for a *blackhole* filter.

IGNORE_RIP_NEIGHBOR statement

Use the IGNORE_RIP_NEIGHBOR statement to specify that RIP routing table broadcasts from the specified gateway are to be ignored. This option can be a RIP input filter.

Syntax

```
►►──Ignore_Rip_Neighbor──IP_address──=──address──────────────────────────────►◄
```

Parameters

`IP_address`
: Specifies the IP address of the gateway from which routing table broadcasts are ignored. For multiple IP addresses, you must repeat the statement for each IP address.

ORIGINATE_RIP_DEFAULT statement

Use the ORIGINATE_RIP_DEFAULT statement to indicate under what conditions RIP supports Default route (destination/mask 0.0.0.0/0.0.0.0) generation.

This statement determines whether or not a default route is considered available by OMPROUTE RIP. The SEND_DEFAULT_ROUTES parameter on the RIP_INTERFACE statement determines whether or not an available default route is advertised by a particular RIP interface.

Syntax

```
                            ┌─Condition=Always──────┐   ┌─Cost=1───────┐
►►──Originate_RIP_Default───┼───────────────────────┼───┼──────────────┼──────────────►
                            └─Condition──=──condition┘   └─Cost──=──cost┘

   ┌─Accept_Default=NO───────┐
►──┼─────────────────────────┼───────────────────────────────────────────────────────►◄
   └─Accept_Default──=──value┘
```

Parameters

`Condition`
: Condition for when RIP is to advertise this router as a default router. Valid values are:

  Always
  : Always originate RIP default. This is the default value.

  OSPF Originate RIP default if OSPF routes are available.

  Never Never advertise this router as a default router.

`Cost`
: Specifies the cost that RIP advertises with the default route that it originates. Valid values are in the range 1 - 16. The default value is 1.

`Accept_Default`
: Specifies whether or not OMPROUTE RIP accepts default routes from inbound RIP packets whose cost is higher than default routes originated by the host.

  Tip: OMPROUTE RIP always accept default routes from inbound RIP packets whose cost is lower than default routes originated by the host.

  A value of **YES** indicates that OMPROUTE RIP replaces this router's originated default route with a default route learned from inbound RIP packets, even if that learned default route has a higher cost than this router's originated default route.

  Results:

  - When **YES** is coded, this router's originated default route is only used if no other default routes are learned from inbound RIP packets. A value of **NO** indicates that OMPROUTE RIP replaces this router's originated default route

with a default route learned from inbound RIP packets only when the learned RIP route has a lower cost than this router's originated default route. This is the default value.
- When this parameter value is NO (either coded or by default), and the other parameters in this statement take their default values (CONDITION=ALWAYS and COST=1), OMPROUTE RIP never accepts default routes learned from RIP packets because it is not possible to learn a RIP route whose cost is less than 1.

RIP_INTERFACE statement

Use the RIP_INTERFACE statement to configure the RIP parameters for each IP interface. Replicated this statement in the configuration file for each IP interface over which RIP operates.

Syntax

```
►►──RIP_Interface──IP_address──=──address──Name──=──interface_name──────────────►

►─Subnet_mask──=──subnet_mask──┬──────────────────────────────────┬─────────────►
                               └─Destination_Addr──=──address─────┘

   ┌─MTU=576──────────┐  ┌─Receive_RIP=YES──────────┐
►──┼──────────────────┼──┼──────────────────────────┼───────────────────────────►
   └─MTU──=──size─────┘  └─Receive_RIP──=──value────┘

   ┌─Receive_Dynamic_Nets=YES─────────┐  ┌─Receive_Dynamic_Subnets=YES─────────┐
►──┼──────────────────────────────────┼──┼─────────────────────────────────────┼──►
   └─Receive_Dynamic_Nets──=──value───┘  └─Receive_Dynamic_Subnets──=──value───┘

   ┌─Receive_Dynamic_Hosts=NO──────────┐
►──┼───────────────────────────────────┼────────────────────────────────────────►
   └─Receive_Dynamic_Hosts──=──value───┘

   ┌──────────────────────────────────────────────────────┐
   ▼                                                      │
►──┴─┬──────────────────────────────────────────────────┬─┴─────────────────────►
     └─filter──=──(filter_type,dest_route,filter_mask)──┘

   ┌─Send_Only=ALL───────────┐  ┌─Send_RIP=YES──────────┐
►──┼─────────────────────────┼──┼───────────────────────┼───────────────────────►
   └─Send_Only──=──(values)──┘  └─Send_RIP──=──value────┘

   ┌─Send_Default_Routes=NO──────────┐  ┌─Send_Net_Routes=YES──────────┐
►──┼─────────────────────────────────┼──┼──────────────────────────────┼────────►
   └─Send_Default_Routes──=──value───┘  └─Send_Net_Routes──=──value────┘
   ┌─Send_Subnet_Routes=YES───────┐ ┌─Send_Static_Routes=NO────────┐
►──┼──────────────────────────────┼─┼──────────────────────────────┼──────────────►
   └─Send_Subnet_Routes──=──value─┘ └─Send_Static_Routes──=──value─┘

   ┌─Send_Host_Routes=NO────────┐ ┌─Send_Poisoned_Reverse_Routes=YES───────┐
►──┼────────────────────────────┼─┼────────────────────────────────────────┼──────►
   └─Send_Host_Routes──=──value─┘ └─Send_Poisoned_Reverse_Routes──=──value─┘

   ┌─In_Metric=1───────────┐ ┌─Out_Metric=0───────────┐ ┌─RipV2=NO────────┐
►──┼───────────────────────┼─┼────────────────────────┼─┼─────────────────┼───────►
   └─In_Metric──=──metric──┘ └─Out_Metric──=──metric──┘ └─RipV2──=──value─┘

   ┌─RipV1_Routes=NO────────┐ ┌─Authentication_Key=nulls──┐
►──┼────────────────────────┼─┼───────────────────────────┼───────────────────────►
   └─RipV1_Routes──=──value─┘ └─Authentication_Key──=──key┘

   ┌────────────────────────┐
   │                        │ ┌─Max_Xmit_Time=120────────┐
►──▼────────────────────────┴─┼──────────────────────────┼────────────────────────►
     └─Neighbor──=──value─┘   └─Max_Xmit_Time──=──time───┘

   ┌─Min_Xmit_Time=0.5─────────┐ ┌─RT_Gain=0.125──────┐
►──┼───────────────────────────┼─┼────────────────────┼───────────────────────────►
   └─Min_Xmit_Time──=──time────┘ └─RT_Gain──=──value──┘

   ┌─Variance_Gain=0.25─────────┐ ┌─Variance_Mult=2─────────┐
►──┼────────────────────────────┼─┼─────────────────────────┼─────────────────────►
   └─Variance_Gain──=──value────┘ └─Variance_Mult──=──mult──┘

   ┌─Delay_Acks=YES────────┐
►──┼───────────────────────┼──────────────────────────────────────────────────────►◄
   └─Delay_Acks──=──value──┘
```

Parameters

IP_address

IP address of interface to be configured for RIP.

The IP address can be a valid IP address that is configured on the system or it can be specified with asterisks (*) as wildcards. The valid wildcard specifications are below. The result of coding a wildcard value are that all configured interfaces whose IP address matches the wildcard are configured as RIP interfaces. Configured interface IP addresses and names are matched against possible wildcards in the order they are in the following example with the name and any matching wildcard being the best match, x.y.z.* being second best, and so on.

```
interface name and any matching wildcard
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.* - Same as ALL
ALL - Same as *.*.*.*
```

Tip: For more information about how wildcard interfaces are parsed, see this Method of assigning interface definitions to stack interfaces (wildcard and explicit): in z/OS Communications Server: IP Configuration Guide.

Because a stack could have a large number of Dynamic VIPAs (DVIPAs) defined, as well as DVIPA ranges, an additional wildcard capability exists on the RIP_INTERFACE statement for use only with DVIPAs. Ranges of DVIPA interfaces can be defined using the subnet mask parameter on the RIP_INTERFACE statement. This mode of definition applies to Dynamic VIPAs defined in the stack with VIPADEFINE, VIPABACKUP, or VIPARANGE. The range defined in this way is all the IP addresses that fall within the subnet defined by the mask and the IP address. When this type of wildcard search is being used, the value of the IP_ADDRESS parameter must be the subnet number of the range. For example, the following defines a range of 6 addresses (9.67.101.9 to 9.67.101.14) that can be used for DVIPA addresses and match any DVIPA interface that falls into the 9.67.101.8/29 subnet:

```
IP_ADDRESS = 9.67.101.8
SUBNET_MASK = 255.255.255.248
```

Alternatively, the following code does not because 9.67.101.17 is an address within the subnet range, not the subnet number itself (that would be 9.67.101.16). This second definition only matches an interface whose home address is 9.67.101.17.

```
IP_ADDRESS= 9.67.101.17
SUBNET_MASK=255.255.255.248
```

Name
The name of the interface. A valid value is any string 1 - 16 characters in length.

Rules:

- If this is not a wildcard interface definition, the name must match the link name that is coded for the corresponding IP address on the HOME statement or the interface name coded for the corresponding IPv4 INTERFACE statement in the TCP/IP profile.
- If this is a wildcard interface definition, then this parameter is used in conjunction with the defined wildcard IP address when searching for definitions to match a stack interface. For more details about this process, see method of assigning interface definitions to stack interfaces (wildcard and explicit): in z/OS Communications Server: IP Configuration Guide.

For Dynamic VIPA (DVIPA), link names are assigned programmatically by the stack when the DVIPA is created; therefore, the name field set on the RIP_INTERFACE statement is ignored by OMPROUTE for DVIPAs.

Subnet_Mask
Subnet mask for the associated interface IP address. For more information, see z/OS Communications Server: IP Configuration Guide. If you configure this interface in the TCP/IP profile using the IPv4 INTERFACE statement and you configure a subnet mask on that statement that does not match the value that you specify on this parameter, OMPROUTE issues message EZZ8164I and uses this subnet mask.

Destination_Addr
IP address of the host at the remote-end of this interface. This parameter is valid only for point-to-point links; it is a required parameter for point-to-point links that cannot receive RIP2 packets (see the description of the RECEIVE_RIP parameter for more information about the level of RIP packets that an interface can receive). If this parameter is not specified for a point-to-point link that can

receive RIP2 packets, a route to the host at the remote end of the interface is not added to the appropriate TCP/IP route tables (main and policy-based tables) until RIP communication is established with that host. A subnet route for the interface is added when OMPROUTE is initialized, whether or not this parameter is specified.

MTU
The maximum transmission unit size that RIP adds to the appropriate routing tables (main and policy-based tables) for routes that use this interface. Valid values are in the range 0 - 65535. If you configure this interface in the TCP/IP profile using the IPv4 INTERFACE statement and you configure an MTU on that statement and the MTU that you configure on that statement does not match the MTU (the configured value or the default value) on this statement, OMPROUTE issues message EZZ8163I and uses the MTU value on this statement.

Tip: See z/OS Communications Server: IP Configuration Guide, in section Maximum transmission unit considerations, for additional information about how TCP/IP uses the MTU to determine the largest size frame to send.

Receive_RIP
Specifies what type of RIP updates are accepted over this interface. Valid values are:

RIP1 Accept only RIP version 1 updates over this interface.

RIP2 Accept only RIP version 2 updates over this interface.

ANY Accept RIP Version 1 and RIP Version 2 updates over this interface.

Rule: If RIP2 authentication is required and this value is coded, unauthenticated RIP1 packets are received over this interface. Also, if RIP2 authentication is not required, authenticated RIP2 packets are *not* be received over this interface, regardless of the value of RIPV2.

YES If RIPV2=YES, then receive only RIP Version 2 updates over this interface. If RIPV2=No, then receive only RIP Version 1 updates over this interface. This is the default value.

NO No RIP packets are received over this interface, regardless of any other filters.

Receive_Dynamic_Nets
Specifies whether or not to learn routes for networks over this interface. If this is not set, only nets explicitly allowed using the Accept_RIP_Route configuration statement is accepted on this interface. Valid values are YES or NO.

Receive_Dynamic_Subnets
Specifies whether or not to learn routes for subnets over this interface. If this is not set, only subnets explicitly allowed using the Accept_RIP_Route configuration statement is accepted on this interface. Valid values are YES or NO.

Receive_Dynamic_Hosts
Specifies whether or not to learn routes for hosts over this interface. If this is not set, only hosts explicitly allowed using the Accept_RIP_Route configuration statement is accepted on this interface. Valid values are YES or NO.

filter
Multiple filter parameters can be coded on a RIP_Interface statement. When specified on the RIP_Interface statement, the filter parameter applies only to

the corresponding RIP interface. The filter statement can also be coded stand-alone in the OMPROUTE configuration file (nosend and noreceive only) to apply to all configured RIP interfaces.

The *filter_type* can be any of the following values:

Value	**Description**
nosend	Specifies that routes matching the *dest_route* and *filter_mask* are not to be broadcast over this interface. This option serves as an RIP output filter.
noreceive	Specifies that routes matching the *dest_route* and *filter_mask* are to be ignored in broadcasts received over this interface. This option serves as an RIP input filter.
send	Specifies that routes matching the *dest_route* and *filter_mask* are to be broadcast over only this interface (or any other RIP interface with an equivalent filter). This option serves as an RIP output filter and can be used for inbound and outbound traffic splitting.
send_cond	Specifies that routes matching the *dest_route* and *filter_mask* are to be broadcast over only this interface when this interface is active (or any other active RIP interface with an equivalent filter). If this interface is inactive, the routes can be broadcast over other interfaces. This option serves as an RIP output filter and can be used for inbound and outbound traffic splitting.
receive	Specifies that routes matching the *dest_route* and *filter_mask* are to be received over only this interface (or any other RIP interface with an equivalent filter). If received over other RIP interfaces, the routes are discarded. This option serves as an RIP input filter.
receive_cond	Specifies that routes matching the *dest_route* and *filter_mask* are to be received over only this interface when this interface is active (or any other active RIP interface with an equivalent filter). If this interface is inactive, the routes can be received over all other active RIP interfaces. This option serves as an RIP input filter.

The *dest_route* specifies the destination route in network, subnetwork, or host format in dotted decimal form. Alternatively, an asterisk (*) can be coded in conjunction with the nosend and noreceive filter types. This serves as a *blackhole* filter that can be used to filter out all routes broadcast or received over an interface. This should be used in conjunction with either additional send or receive filters to allow only certain routes to be received, or advertised over an interface or set of interfaces.

Tip: If the blackhole nosend filter is used, it does not filter out the sending of the default route when the Originate_RIP_Default statement is also configured.

The *filter_mask* specifies the filter mask in dotted decimal form. If not coded, the default filter mask is 255.255.255.255, meaning apply the filter to the *dest route* as coded. Coding the filter mask has no meaning and is not valid if the *dest route* is coded as an asterisk (*) for a *blackhole* filter.

`Send_Only`
Specifies broadcast restrictions. Multiple values can be coded by separating the values with commas, unless ALL is coded. The valid values are:

ALL Specifies no broadcast restrictions.

VIRTUAL
Broadcasts virtual IP addresses.

DEFAULT
Broadcasts the default route.

DIRECT
Broadcasts direct routes.

TRIGGERED
Only broadcasts routes when requested or when a route becomes inactive (metric 16).

VIRTUAL, DEFAULT, and DIRECT are Or'd together to determine what should be broadcast. Thus, coding SEND_ONLY=(VIRTUAL, DEFAULT) broadcasts virtual IP addresses and the default route.

Restriction: When ALL is coded it must not be enclosed within parentheses. When any of the other possible values are coded, they must be enclosed within parentheses.

When specified on the RIP_Interface statement, the Send_Only parameter applies only to the corresponding RIP interface. The Send_Only statement can also be coded stand-alone in the OMPROUTE configuration file to apply to all RIP interfaces.

`Send_RIP`
Specifies whether or not RIP advertisements are broadcast over this interface. Valid values are YES or NO.

`Send_Default_Routes`
Advertise the default route (destination 0.0.0.0), if it is available, in RIP responses sent from this IP source address. Valid values are YES or NO.

Restriction: If DEFAULT is coded on the Send_Only parameter or the stand-alone Send_Only statement, the Send_Default_Routes parameter is ignored and is set to YES.

`Send_Net_Routes`
Advertise all network level routes in RIP responses sent from this IP address. Valid values are YES or NO.

`Send_Subnet_Routes`
Advertise appropriate subnet-level routes in RIP responses sent from this IP address. Valid values are YES or NO.

In this context an appropriate subnet is one that meets RFC 1058 subnet advertisement constraints as follows:
- Natural Net must be the same as the IP source's natural net.
- Subnet mask must be the same.

`Send_Static_Routes`
Advertise static and direct routes in RIP responses sent from this IP source address. Split horizon is applied; that is, static routes configured over an interface are not included in RIP responses sent from that interface. Valid values are YES or NO.

Send_Host_Routes
Advertise host routes in RIP responses sent from this IP source address. In this context, a host route is one with a mask of 255.255.255.255. Valid values are YES or NO.

Send_Poisoned_Reverse_Routes
Advertise poisoned reverse routes over the interface corresponding to the next hop. A poison reverse route is one with an infinite metric (16). Valid values are YES or NO. If NO is specified, OMPROUTE still uses split horizon.

In_Metric
Specifies the value of the metric to be added to RIP routes that are received over this interface before the routes are installed in the appropriate routing tables (main and policy-based tables). Valid values are in the range 1 - 15.

Out_Metric
Specifies the value of the metric to be added to RIP routes advertised over this interface. Valid values are in the range 0 - 15.

RipV2
Enables RIP V2 packets to be sent on this link. Valid values are YES or NO. If YES, all RIP packets sent on this link are RIPV2. If NO, all RIP packets sent on this link are RIPV1. See the RECEIVE_RIP description in this list for information about configuring the level of RIP packets that can be received on this link.

RipV1_Routes
Specifies whether RIP V1 routes should be advertised on this RIP V2 link. Valid values are YES or NO.

Authentication_Key
RIP V2 authentication key. Only used for RIP V2 packets. Coding this key does not prevent reception of unauthenticated RIP V1 packets. To ensure that only authenticated RIP packets can be received over this interface, code RECEIVE_RIP=RIP2 in addition to this parameter. Valid values are any alphanumeric string from code page 1047 up to 16 characters in length coded within double quotation marks, or any hexadecimal string which begins with 0x.

Rules:
- If the value is entered in characters (rather than the hexadecimal string), that value is case sensitive.
- If an authentication key is not provided, authenticated RIP V2 packets are not received, even if RECEIVE_RIP=ANY.

Neighbor
Specifies the IP address of a single neighboring router. Multiple Neighbor parameters can be coded on a RIP_Interface statement to specify each adjacent RIP router. Use the Neighbor parameter when the interface is not point-to-point, does not support broadcast, and either does not support multicast or is using RIP version 1. Examples of interface types for which the Neighbor parameter must be used are:
- Hyperchannel
- ATM
- For RIP V1, OSA QDIO that does not have the IPBCAST keyword specified on the LINK or INTERFACE statement in the TCPIP PROFILE

Guideline: Do not define neighbors on multicast-capable media if this interface supports RIP V2, or broadcast-capable media for interfaces that support RIP V1 or RIP V2. If you define neighbors on these media, OMPROUTE is able to communicate RIP information *only* with those neighbors that are defined (it does not learn about any additional neighbors).

Retransmit Parameters

The following parameters are used by OMPROUTE to set values in the routes that use this interface; the values are added to the TCP/IP route tables. The values affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a certain number of retransmits, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times packets are retransmitted before the connection times out differs for initial connection establishment and for data packets. For initial connection establishment, the initial time interval is set at approximately 3 seconds, and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval and are retransmitted 15 times before the connection is timed out. All of the following parameters affect the data packet retransmission algorithm. Only the Min_Xmit_Time parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

Max_Xmit_Time
: Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to time out. Specifying Max_Xmit_Time assures that the interval time never exceeds the specified limit. The minimum value that can be specified for Max_Xmit_Time is 0. The maximum is 999.990. The default is 120 seconds. This parameter affects the initial connection establishment retransmission timeout for all APIs, except the Pascal API (TcpOpen), that are using the socket connect function.

Min_Xmit_Time
: Sets a minimum retransmit interval. Increasing this value might increase the amount of time it takes for TCP to time out a connection. The minimum value that can be specified for Min_Xmit_Time is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

RT_Gain
: This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet's RTT has on the average. The minimum value that can be specified for RT_Gain is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

Variance_Gain
: This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum

value that can be specified for Variance_Gain is 0. The maximum value is 1.0. The default is 0.25. This parameter does not affect initial connection retransmission.

Variance_Mult
This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT has on calculating the retransmission interval. The minimum value that can be specified for Variance_Mult is 0. The maximum value is 999.990. The default is 2. This parameter does not affect initial connection retransmission.

Delay_Acks
The delay acknowledgments value that is added to the routing tables for routes that use this interface. Specify YES to delay transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specify NO to return acknowledgments immediately when a packet is received with the PUSH bit on in the TCP header. This parameter affects only connections that use the routes associated with this interface.

Even if you specify YES, you can override the delay acknowledgments behavior by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements. A value of NO can override the specification of the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

Valid values are YES and NO. The default value is YES.

SEND_ONLY statement

The SEND_ONLY statement can be coded stand-alone in the OMPROUTE configuration file to apply to all RIP interfaces.

Syntax

```
   ┌─Send_Only=ALL────────────┐
►►─┴─Send_Only──=──(values)───┴──────────────────────────────────────────►◄
```

Parameters

(values)
Specifies broadcast restrictions. Multiple values can be coded by separating the values with commas, unless ALL is coded. The valid values are:

ALL Specifies no broadcast restrictions.

VIRTUAL
Broadcasts virtual IP addresses.

DEFAULT
Broadcasts the default route.

DIRECT
Broadcasts direct routes.

TRIGGERED
Only broadcast routes when requested or when a route becomes inactive (metric 16).

VIRTUAL, DEFAULT, and DIRECT are OR'd together to determine what should be broadcast. Thus, coding SEND_ONLY=(VIRTUAL, DEFAULT) broadcasts virtual IP addresses and the default route. When ALL is coded, it must not be enclosed within parentheses. When any of the other possible values are coded, they must be enclosed within parentheses.

When specified on the SEND_ONLY statement in the OMPROUTE configuration file, it applies to all RIP_Interfaces. The SEND_ONLY parameter can also be coded on the RIP_INTERFACE statement. When specified on the RIP_INTERFACE statement, the SEND_ONLY parameter applies only to the corresponding RIP_Interface.

IPv6 OSPF configuration statements

This topic contains descriptions of the following IPv6 OSPF configuration statements:

- IPv6_AREA
- IPv6_AS_BOUNDARY_ROUTING
- IPv6_OSPF
- IPv6_OSPF_INTERFACE
- IPv6_RANGE
- IPv6_VIRTUAL _LINK

See z/OS Communications Server: IP System Administrator's Commands for information about how to display configuration information.

IPv6_AREA statement

Use the IPv6_AREA statement to set the parameters for an IPv6 OSPF area. If no areas are defined, OMPROUTE assumes that all the router's directly attached networks belong to the backbone area (area ID 0.0.0.0).

Syntax

```
                                                 ┌─Stub_Area=NO──────────┐
►►──IPv6_Area──Area_Number──=──ospf_area_address─┼───────────────────────┼──────────────►
                                                 └─Stub_Area──=──value───┘

  ┌─Stub_Default_Cost=1────────┐   ┌─Import_Prefixes=YES────────┐
►─┼────────────────────────────┼───┼────────────────────────────┼────────────────────►◄
  └─Stub_Default_Cost──=──cost─┘   └─Import_Prefixes──=──value──┘
```

Parameters

Area_Number
The 32–bit OSPF area number in dotted decimal.

Stub_Area
Specifies whether this area is a stub area or not. Valid values are YES or NO.

Restrictions: If you specify Stub_area = YES, the area does not receive any AS external link advertisements, reducing the size of your database and decreasing memory usage for routers in the stub area. The following restrictions apply:

- You cannot configure virtual links through a stub area.
- You cannot configure a router within the stub area as an AS boundary router.

- You cannot configure the backbone as a stub area.

External routing in stub areas is based on a default route. Each area border router attaching to a stub area originates a default route for this purpose. The cost of this default route is also configurable with the IPv6_AREA statement.

`Stub_Default_Cost`
The cost that an OMPROUTE area border router associates with the default route that it generates into the stub area. Valid values are in the range 1 - 16 777 215.

`Import_Prefixes`
If this area is a stub area, indicates whether prefixes from neighboring areas are imported. Valid values are YES or NO.

Tip: A stub area with Import_Prefixes set to NO is commonly referred to in RFCs and other standards documentation as a Totally Stubby Area.

IPv6_AS_BOUNDARY_ROUTING statement

Use the IPv6_AS_BOUNDARY_ROUTING statement to enable the AS boundary routing capability, which allows you to import routes learned from other methods (IPv6 RIP, statically configured, or direct routes) into the IPv6 OSPF domain. All routes are imported as either Type 1 or Type 2 external routes, depending on what was coded on the Comparison statement. The metric type used when importing routes determines how the imported cost is viewed by the IPv6 OSPF domain. When comparing Type 2 metrics, only the external cost is considered in selecting the best route. When comparing Type 1 metrics, the external and internal costs of the route are combined before making the comparison.

Requirement: This statement must be coded even if the only route you want to import is the default route (prefix length 0).

Syntax

```
                            ┌─Import_RIP_Routes=No──────┐
►►──IPv6_AS_Boundary_Routing──┼───────────────────────────┼──────────────────────────►
                            └─Import_RIP_Routes──=──value─┘

  ┌─Import_Static_Routes=No──────┐ ┌─Import_Direct_Routes=No──────┐
►─┼──────────────────────────────┼─┼──────────────────────────────┼──────────────────►
  └─Import_Static_Routes──=──value─┘ └─Import_Direct_Routes──=──value─┘

  ┌─Import_Router_Advertisement_Routes=No───┐
►─┼─────────────────────────────────────────┼────────────────────────────────────────►
  └─Import_Router_Advertisement_Routes──value─┘

  ┌─Originate_Default_Route=No──────┐ ┌─Originate_as_Type=2──────┐
►─┼─────────────────────────────────┼─┼──────────────────────────┼───────────────────►
  └─Originate_Default_Route──=──value─┘ └─Originate_as_Type──=──type─┘

  ┌─Default_Route_Cost=1──────┐ ┌─Learn_Default_Route=NO──────┐
►─┼───────────────────────────┼─┼─────────────────────────────┼──────────────────────►
  └─Default_Route_Cost──=──cost─┘ └─Learn_Default_Route──=──value─┘

►─┬───────────────────────────────────────────┬──────────────────────────────────────►◄
  └─Default_Forwarding_Address──=──ip-address─┘
```

Parameters

`Import_RIP_Routes`
Specifies whether routes learned by IPv6 RIP are imported into the IPv6 OSPF routing domain. Valid values are YES or NO.

`Import_Static_Routes`
Specifies whether static routes (routes defined to the TCP/IP stack using the BEGINROUTES or GATEWAY statements) are imported into the IPv6 OSPF routing domain. Valid values are YES or NO.

`Import_Direct_Routes`
Specifies whether IPv6 direct routes are imported into the IPv6 OSPF routing domain. Valid values are YES or NO.

`Import_Router_Advertisement_Routes`
Specifies whether routes learned by the TCP/IP stack from IPv6 Router Advertisements are imported into the IPv6 OSPF routing domain. Valid values are YES and NO.

Tip: If a router is advertising a route into the OSPF domain on a link LSA, it is considered an OSPF internal route, regardless of whether or not it is also being advertised in an IPv6 Router Advertisement. Therefore, this parameter only controls routes that are only advertised by routers in IPv6 Router Advertisements.

`Originate_Default_Route`
Specifies whether or not this host originates an AS External default route into the IPv6 OSPF domain. If YES and Default_Forwarding_Address is not also coded (or is coded to ::), this host advertises itself as a default router. Valid values are YES or NO.

`Originate_as_Type`
Specifies the external type assigned to the default route originated by this host if Originate_Default_Route is YES. Valid values are 1 or 2.

Tip: See the comparison parameter in "IPv6_OSPF statement" on page 521 for more information about external route types.

`Default_Route_Cost`
Specifies the cost that IPv6 OSPF associates with the default route originated by this host if Originate_Default_Route is YES. Valid values are in the range 0 - 16 777 215.

`Learn_Default_Route`
Specifies whether IPv6 OSPF learns default routes from inbound packets when their cost is equal to or higher than the cost of the default route originated by this host. Valid values are YES or NO. If this parameter is set to NO, then only default routes with lower cost than the one originated by this host are learned.

`Default_Forwarding_Address`
If Orignate_Default_Route is YES, this optional parameter can be used to specify that this host should originate a default route on behalf of a different router. This parameter is not needed if this host is to advertise itself as the default router. It should only be used when the default router is another router that this host can route to, which is not capable of advertising an IPv6 OSPF default route on its own behalf. In that case, this parameter should be set to a reachable IP address on the other router.

Restriction: This address must be reachable using an OSPF intra-area or inter-area route (labelled as SPF or SPIA in the RT6TABLE display, or labelled as DIR but using an OSPF interface). This route could be a host, prefix, or default route. If no eligible route is found, the forwarding address is not included in the advertisements generated by this statement.

IPv6_OSPF statement

Use the IPv6_OSPF statement to specify various parameters that apply globally to IPv6 OSPF, either to all interfaces or to the overall IPv6 OSPF autonomous system.

Syntax

```
                                              ┌─Comparison──=──Type2─┐
►►──IPv6_OSPF──┬───────────────────────┬──────┼──────────────────────┼────────────────────────────►
               └─RouterID──=──value────┘      └─Comparison──=──value─┘

   ┌─Demand_Circuit──=──YES────┐   ┌─DR_Max_Adj_Attempt──=──0─────┐
►──┼───────────────────────────┼───┼──────────────────────────────┼───────────────────────────────►
   └─Demand_Circuit──=──value──┘   └─DR_Max_Adj_Attempt──=──value─┘

   ┌─Instance──=──0──────┐
►──┼─────────────────────┼────────────────────────────────────────────────────────────────────────►◄
   └─Instance──=──value──┘
```

Parameters

RouterID
Every router in an IPv6 OSPF routing domain must be assigned a unique 32-bit router ID.

The value used for the IPv6 OSPF router ID is chosen as follows:

- If this router ID is configured, the value configured is used as the IPv6 OSPF router ID.

  Rule: The reserved 0.0.0.0 address cannot be used as a router ID.
- If this router ID is not configured and IPv4 OSPF is also active on OMPROUTE, then the IPv4 Router ID value is also be used for IPv6.

Valid values are any 32-bit value, in dotted decimal format (in other words, specified as an IPv4-style IP address).

Restriction: If IPv4 OSPF is NOT active, then RouterID is a required configuration parameter.

Comparison
Tells OMPROUTE where external routes fit in the IPv6 OSPF hierarchy. IPv6 OSPF supports two types of external metrics. Type1 external metrics are equivalent to the link state metric. Type2 external metrics are greater than the cost of any path internal to the AS. Use of Type2 external metrics assumes that routing between autonomous systems is the major cost of routing a packet, and eliminates the need for conversion of external costs to internal link state metrics. Valid values are Type1 (or 1) or Type2 (or 2).

For more information about the COMPARISON configuration parameter, see z/OS Communications Server: IP Configuration Guide.

Demand_Circuit
Global demand circuit setting. Coding YES enables demand circuits for IPv6 OSPF. Demand circuit parameters can then be coded on the IPv6_OSPF_Interface statement. Valid values are Yes or No.

DR_Max_Adj_Attempt
Specifies the maximum number of adjacency attempts to be used for reporting and controlling futile neighbor state loops. After the adjacency attempt count for a neighboring designated router reaches the threshold, an informational message is issued to report the problem. If a redundant interface is available that can reach the neighbor, adjacency formation is attempted over that interface. An informational message is issued to report the interface switch and the adjacency formation attempt. Valid values are in the range 0 - 100. The value 0 indicates infinite retries.

For information about futile neighbor state loops, see the network design considerations information in z/OS Communications Server: IP Configuration Guide. For the types of interfaces supporting the futile neighbor state loop detection for OSPF, see "Interfaces supported by OMPROUTE" on page 548.

Instance
Provides the default instance number for OMPROUTE. OMPROUTE supports only one instance of IPv6 OSPF on a link, and this parameter specifies the default value for all IPv6 OSPF interfaces. This value can be overriden on individual IPv6_OSPF_Interface statements. Valid values are any integer from 0 - 255.

IPv6_OSPF_INTERFACE statement

Use the IPv6_OSPF_INTERFACE statement to set the IPv6 OSPF parameters for the TCP/IP network interfaces. Replicate this statement in the configuration file for each IPv6 interface over which OSPF operates.

Syntax

```
►►──IPv6_OSPF_Interface──Name──=──interface_name──────────────────────────────►


   ┌─────────────────────────────────┐  ┌─Instance──=──0──────┐
   ▼                                 │  │                     │
►──┴──┬───────────────────────────┬──┴──┼─────────────────────┼────────────────►
      └─Prefix──=──prefix/prefixlen─┘    └─Instance──=──value──┘

   ┌─Attaches_To_Area──=──0.0.0.0─┐  ┌─Transmission_Delay=1──────────┐
►──┼──────────────────────────────┼──┼───────────────────────────────┼─────────►
   └─Attaches_To_Area──=──area────┘  └─Transmission_Delay──=──delay──┘

   ┌─Retransmission_Interval=5──────────────┐  ┌─Router_Priority=1─────────────┐
►──┼────────────────────────────────────────┼──┼───────────────────────────────┼►
   └─Retransmission_Interval──=──frequency──┘  └─Router_Priority──=──priority──┘

   ┌─Hello_Interval=10──────────────┐  ┌─DB_Exchange_Interval=40───────────────┐
►──┼────────────────────────────────┼──┼───────────────────────────────────────┼─►
   └─Hello_Interval──=──interval────┘  └─DB_Exchange_Interval──=──interval─────┘
   ┌─Dead_Router_Interval=40─────────────┐ ┌─Cost=1─────────┐
►──┼─────────────────────────────────────┼─┼────────────────┼──────────────────────────────►
   └─Dead_Router_Interval──=──interval───┘ └─Cost──=──cost──┘

   ┌─Demand_Circuit=no─────────┐ ┌─Hello_Suppression=Allow─────────┐
►──┼───────────────────────────┼─┼─────────────────────────────────┼──────────────────────►
   └─Demand_Circuit──=──value──┘ └─Hello_Suppression──=──value─────┘

   ┌─PP_Poll_Interval=60──────────────┐ ┌─Parallel_OSPF=Backup──────┐
►──┼──────────────────────────────────┼─┼───────────────────────────┼─────────────────────►
   └─PP_Poll_Interval──=──interval────┘ └─Parallel_OSPF──=──value───┘

   ┌─Max_Xmit_Time=120────────┐ ┌─RT_Gain=0.125─────────┐
►──┼──────────────────────────┼─┼───────────────────────┼─────────────────────────────────►
   └─Max_Xmit_Time──=──time───┘ └─RT_Gain──=──value─────┘

   ┌─Min_Xmit_Time=0.5────────┐ ┌─Variance_Gain=0.25──────────┐
►──┼──────────────────────────┼─┼─────────────────────────────┼───────────────────────────►
   └─Min_Xmit_Time──=──time───┘ └─Variance_Gain──=──value─────┘

   ┌─Variance_Mult=2──────────┐ ┌─Delay_Acks=YES────────┐
►──┼──────────────────────────┼─┼───────────────────────┼────────────────────────────────►◄
   └─Variance_Mult──=──mult───┘ └─Delay_Acks──=──value──┘
```

Parameters

Name

The name of the interface.

This name must match the interface name coded on the INTERFACE or VIPADYNAMIC statement in the TCP/IP profile. Valid values are any character string of 1 - 16 characters in length. Wildcard names (terminating in *) can be coded. For example, OSAQDIO* would match stack interfaces named OSAQDIO1, OSAQDIO2, OSAQDIOABC, and so forth.

Tips:

- For more information about how wildcard interfaces are parsed, see the step about defining IPv6 interfaces in z/OS Communications Server: IP Configuration Guide.
- For the names to use when defining IPv6 dynamic XCF interfaces, see the step about defining IPv6 interfaces in z/OS Communications Server: IP Configuration Guide.

Prefix

Specifies a prefix that is on the link to which the interface attaches. For each configured Prefix parameter, OMPROUTE adds a direct route to the prefix identified by the first *prefixlen* bits of *prefix*. Valid values for *prefix* are any colon-hexadecimal IPv6 address. Valid values for *prefixlen* are any integer value from 1 - 127. The prefix identified by the first *prefixlen* bits of *prefix* must not be a multicast prefix, a link-local prefix, or all zeros.

Guideline: If routers on the link are advertising prefixes using either IPv6 OSPF or IPv6 Router Discovery, prefixes being advertised as on-link by the routers should not be configured using this keyword. However, if IPv6 Router Discovery or IPv6 OSPF is not in use by the routers on the link or there is a need to supplement the list of prefixes being advertised as on-link by the routers, this keyword can be used. If the prefix is configured using this keyword and is also advertised by a router as being on-link, the route in the TCPIP stack's route table is the route added by OMPROUTE as a result of this keyword being specified. Any route for the same prefix that is learned from IPv6 OSPF or Router Discovery is ignored as long as the OMPROUTE-configured route exists.

`Instance`
Specifies the IPv6 protocol instance number for this interface. This value should be the same as the instance value of other IPv6 OSPF hosts or routers that OMPROUTE communicates with on this link. This value is set on all outgoing IPv6 OSPF packets, and all incoming IPv6 OSPF packets whose instance value does not match the value coded for this interface are ignored. This permits multiple instances of OSPF to be run on this link. OMPROUTE supports only one instance per link; however, by coding this parameter, OMPROUTE can interact with other routers that can support multiple instances. This value defaults to the value coded on the Instance parameter of the IPv6_OSPF configuration statement. If that value is not coded, the default is 0. Valid values are in the range 1 - 255.

`Attaches_To_Area`
IPv6 OSPF area to which this interface attaches. Valid values are 0.0.0.0 (the backbone), or any area defined by the IPv6_AREA statement.

`Retransmission_Interval`
Sets the frequency (in seconds) of retransmitting link-state update packets, link-state request packets, and database description packets. Valid values are in the range 1 - 65535 seconds.

Guideline: If this parameter is set too low, needless retransmissions occur that could affect performance and interfere with neighbor adjacency establishment. It should be set to a higher value for a slower machine.

`Transmission_Delay`
This parameter is an estimate of the number of seconds that it takes to transmit link-state information over the interface. Each link-state advertisement has a finite lifetime of 1 hour. As each link-state advertisement is sent out from this interface, it is aged by this configured transmission delay. Valid values are in the range 1 - 65535 seconds.

`Router_Priority`
This value is used for multiaccess networks to elect the designated router, with the highest priority router being elected. Valid values are in the range 0 - 255. A value of 0 indicates that OMPROUTE cannot become designated router.

A value of 1 indicates the lowest possible eligible priority and a value of 255 indicates the highest possible priority. A value of 0 indicates that OMPROUTE is not eligible to be a designated router on this link.

`Hello_Interval`
This parameter defines the number of seconds between IPv6 OSPF Hello packets being sent out this interface. This value must be the same for all routers attached to a common link. Valid values are in the range 1 - 255 seconds.

`DB_Exchange_Interval`
The interval in seconds that the database exchange process cannot exceed. If

the interval elapses, the procedure is restarted. This value must be larger than the Hello_Interval. If no value is specified, the DB_Exchange_Interval is set to the Dead_Router_Interval. Valid values are 2 through 65535.

`Dead_Router_Interval`
The interval in seconds, after not having received an IPv6 OSPF Hello, that the neighbor is declared to be down. This value must be larger than the Hello_Interval. Setting this value too close to the Hello_Interval can result in the collapse of adjacencies. A value of 4*Hello_Interval is preferred. This value must be the same for all routers attached to a common link. Valid values are 2 to 65535.

`Cost`
The OSPF cost for this interface. The cost is used to determine the shortest path to a destination. Valid values are in the range 1 - 65535.

`Demand_Circuit`
This parameter, when coded with YES, causes Link State Advertisements (LSAs) to not be periodically refreshed over this interface. Only LSAs with real changes are advertised. In addition, coding this parameter to YES causes LSAs flooded over this interface to never age out. Valid values are YES or NO. For more information about the Demand_Circuit=YES and related topics, such as handling high cost links, see z/OS Communications Server: IP Configuration Guide.

`Hello_Suppression`
This parameter is meaningful only for demand circuits. This parameter allows you to configure the interface to request Hello_Suppression. This parameter is used only on point-to-point and point-to-multipoint interfaces. Valid values are ALLOW, REQUEST, or DISABLE.

If either or both sides specify DISABLE, Hello_Suppression is disabled. If both specify ALLOW, Hello_Suppression is disabled. If one specifies ALLOW and the other REQUEST, or if both specify REQUEST, Hello_Suppression is enabled.

`PP_Poll_Interval`
This parameter specifies the interval (in seconds) that OMPROUTE should use when attempting to contact a neighbor to reestablish a neighbor relationship when the relationship has failed, but the interface is still available. This parameter is meaningful only if Demand_Circuit is coded YES and Hello_Supression has been enabled. Valid values are in the range 0 - 65535.

`Parallel_OSPF`
This parameter designates whether the IPv6 OSPF interface is primary or backup when more than one IPv6 OSPF interface is defined to the same link. Only one of these interfaces can be configured as primary, meaning that it is the interface to carry the IPv6 OSPF protocol traffic between OMPROUTE and the subnet. Failure of the primary interface results in automatic switching of OSPF traffic to one of the backup interfaces. If the primary interface is later reactivated, IPv6 OSPF traffic is not automatically switched back from the backup interface to the primary interface. If you want to switch OSPF traffic back to the primary interface, the backup interface must be stopped. If none of the interfaces to the common subnet are configured as primary, a primary interface is selected by OMPROUTE. Valid values are Backup and Primary.

Tip: For IPv6, OMPROUTE considers two interfaces to be on the same link if they have any prefixes in common.

Retransmit Parameters

The following parameters are used by OMPROUTE to set values in the routes added to theTCP/IP route table, which use this interface. The values affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a certain number of retransmits, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times packets are retransmitted before the connection times out differs for initial connection establishment and for data packets. For initial connection establishment, the initial time interval is set at approximately 3 seconds, and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval and are retransmitted 15 times before the connection is timed out. All of the following parameters affect the data packet retransmission algorithm. Only the Min_Xmit_time parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

Max_Xmit_Time
: Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to time out. Specifying Max_Xmit_Time assures that the interval time never exceeds the specified limit. The minimum value that can be specified for Max_Xmit_Time is 0. The maximum is 999.990. The default is 120 seconds. This parameter affects the initial connection establishment retransmission timeout for all APIs, except the Pascal API (TcpOpen), that are using the socket connect function.

Min_Xmit_Time
: Sets a minimum retransmit interval. Increasing this value might increase the amount of time it takes for TCP to time out a connection. The minimum value that can be specified for Min_Xmit_Time is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

RT_Gain
: This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet's RTT has on the average. The minimum value that can be specified for RT_Gain is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

Variance_Gain
: This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum value that can be specified for Variance_Gain is 0. The maximum value is 1.0. The default is 0.25. This parameter does not affect initial connection retransmission.

Variance_Mult
: This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT

has on calculating the retransmission interval. The minimum value that can be specified for Variance_Mult is 0. The maximum value is 99.990. The default is 2. This parameter does not affect initial connection retransmission.

Delay_Acks
The delay acknowledgments value that is added to the routing tables for routes that use this interface. Specify YES to delay transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specify NO to return acknowledgments immediately when a packet is received with the PUSH bit on in the TCP header. This parameter affects only connections that use the routes associated with this interface.

Even if you specify YES, you can override the delay acknowledgments behavior can be overridden by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements. A value of NO can override the specification of the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

Valid values are YES and NO. The default value is YES.

IPv6_RANGE statement

Use the IPv6_RANGE statements to ad ranges to IPv6 OSPF areas. External to the area, a single route is advertised for each address range. For example, if an IPv6 OSPF area were to consist of the prefix 2001:0db8:1:2::/64, all addresses falling within that prefix would be defined as consisting of a single address range. The address range would be specified as an address of 2001:0db8:1:2:: together with a prefix length of 64. Outside of the area, all addresses that fall within that prefix would be advertised as a single route to prefix 2001:0db8:1:2::/64.

Ranges can be defined to control which routes are advertised external to an area.

There are two choices:
- When IPv6 OSPF is configured to advertise the range, a single interarea route is advertised for the range if at least one component route of the range is active within the area.
- When IPv6 OSPF is configured not to advertise the range, no interarea prefix routes are advertised for routes that fall within the range. Ranges cannot be used for areas that serve as transit areas for virtual links. Also, when ranges are defined for an area, IPv6 OSPF does not function correctly if the area is partitioned but is connected by the backbone.

Ranges cannot be used for areas that serve as transit areas for virtual links. Also, when ranges are defined for an area, OSPF does not function correctly if the area is partitioned but is connected by the backbone.

Syntax

```
                                            ┌─Area_Number=0.0.0.0───┐
►►──IPv6_Range──Prefix──=──prefix/prefixlen─┼───────────────────────┼──────────►
                                            └─Area_Number──=──area──┘

   ┌─Advertise=YES─────────┐
►──┼───────────────────────┼──────────────────────────────────────────────────►◄
   └─Advertise──=──value───┘
```

Parameters

Prefix
Common prefix of IP addresses in this range, with the prefix length.

Area_Number
Area number for which to add this range. Valid values are any defined areas.

Advertise
Determines whether this range is advertised to other areas. Valid values are YES or NO.

IPv6_VIRTUAL_LINK statement

Use the IPv6_VIRTUAL_LINK statement to configure a virtual link between two area border routers. To maintain backbone connectivity you must have all of your backbone routers interconnected either by permanent or virtual links. Virtual links are considered to be separate router interfaces connecting to the backbone area. Therefore, you are asked to specify many of the interface parameters when configuring a virtual link.

Virtual links can be configured between any two backbone routers that have an interface to a common nonbackbone, nonstub area. Virtual links are used to maintain backbone connectivity and must be configured at both endpoints.

Tip: Do not confuse OSPF virtual links with Virtual IP Address support (VIPA).

Syntax

```
►►──IPv6_Virtual_Link──Virtual_Endpoint_RouterID──=──id─────────────────────────────►


                              ┌─Retransmission_Interval=10───────────────┐
►─Links_Transit_Area──=──area─┼──────────────────────────────────────────┼──────────►
                              └─Retransmission_Interval──=──frequency────┘

  ┌─Transmission_Delay=5───────────┐   ┌─Hello_Interval=30──────────────┐
►─┼────────────────────────────────┼───┼────────────────────────────────┼───────────►
  └─Transmission_Delay──=──delay───┘   └─Hello_Interval──=──interval────┘

  ┌─Dead_Router_Interval=180──────────────┐   ┌─DB_Exchange_Interval=180──────────────┐
►─┼───────────────────────────────────────┼───┼───────────────────────────────────────┼──►◄
  └─Dead_Router_Interval──=──interval─────┘   └─DB_Exchange_Interval──=──interval─────┘
```

Parameters

Virtual_Endpoint_RouterID
32-bit IPv6 OSPF router ID of the virtual neighbor (other endpoint), specified in dotted-decimal notation.

Links_Transit_Area
This is the nonbackbone, nonstub area through which the virtual link is configured. Virtual links can be configured between any two area border routers that have an interface to a common nonbackbone and nonstub area. Virtual links must be configured in each of the link's two endpoints. Valid values are 0.0.0.1 - 255.255.255.255.

`Retransmission_Interval`
Sets the frequency (in seconds) of retransmitting link-state update packets, link-state request packets, and database description packets. Valid values are from in the range 1 - 65 535 seconds.

Guideline: If this parameter is set too low, needless retransmissions occur that could affect performance and interfere with neighbor adjacency establishment. It should be set to a higher value for a slower machine.

`Transmission_Delay`
This parameter is an estimate of the number of seconds that it takes to transmit link-state information over the virtual link. Each link-state advertisement has a finite lifetime of one hour. As each link-state advertisement is sent out from this virtual link, it is aged by this configured transmission delay. Valid values are in the range 1 - 65 535 seconds.

`Hello_Interval`
This parameter defines the number of seconds between OSPF Hello packets being sent out from this virtual link. Valid values are in the range 1 - 255 seconds. The Hello_Interval should be set higher than the same value used on the intervening, actual IPv6 OSPF interfaces.

`Dead_Router_Interval`
The interval in seconds, after not having received an OSPF Hello, that the neighbor is declared to be down. This value must be larger than the Hello_Interval. Valid values are 2 - 65 535. The dead router interval should be set higher than the same value used on the intervening, actual, IPv6 OSPF interfaces.

`DB_Exchange_Interval`
The interval in seconds that the database exchange process cannot exceed. If the interval elapses, the procedure is restarted. This value must be larger than the Hello_Interval. If no value is specified, the DB_Exchange_Interval is set to the Dead_Router_Interval. Valid values are 2 - 65 535.

IPv6 RIP configuration statements

This topic contains descriptions of the following IPv6 RIP configuration statements:
- IPV6_ACCEPT_RIP_ROUTE
- IPV6_RIP_FILTER
- IPV6_IGNORE_RIP_NEIGHBOR
- IPV6_ORIGINATE_RIP_DEFAULT
- IPV6_RIP_INTERFACE
- IPV6_RIP_SEND_ONLY

IPv6_ACCEPT_RIP_ROUTE statement

Allows a prefix or host route to be accepted independent of whether the interface it was received on has the corresponding reception parameter enabled (prefix or host). Routes added in this manner can be thought of as a list of exception conditions.

Syntax

```
►►──IPv6_Accept_RIP_Route──IP-address──=──address──────────────────────────────►◄
```

Parameters

IP_address
Destination route to be unconditionally accepted, specified in colon-hexadecimal format.

IPv6_RIP_FILTER statement

Use the IPv6_RIP_FILTER statement to allow for the specification of routes that are not to be sent or received over IPv6 RIP interfaces. The IPv6_RIP_Filter statement can be coded stand-alone in the OMPROUTE configuration file (nosend and noreceive only) to apply to all configured IPv6 RIP interfaces.

Syntax

```
►►──IPv6_RIP_Filter=(type,dest/prefix_len)──────────────────────────►◄
```

Parameters

type
The type can be any of the following values:

nosend
Specifies that routes matching the dest and prefix_len are not to be sent over IPv6 RIP interfaces. This option serves as an IPv6 RIP output filter.

noreceive
Specifies that routes matching the dest and prefix_len are to be ignored in messages received over IPv6 RIP interfaces. This option serves as an IPv6 RIP input filter.

dest
The dest specifies the destination route in colon-hexadecimal format. Alternatively, an asterisk (*), which matches *any* IPv6 destination, can be coded to filter out all routes sent or received over an interface. The use of the asterisk is also referred to as a blackhole filter. This should be used in conjunction with either additional send or receive filters to allow only certain routes to be received, or advertised over an interface or set of interfaces.

Tip: The blackhole nosend filter does not filter out the sending of the default route when the IPv6_Originate_RIP_Default statement is also configured.

prefix_len
The prefix_len specifies the number of significant bits in the destination to be filtered. If not coded, the default prefix_len is 128, meaning apply the filter to the dest as coded. Coding the prefix_len has no meaning and is not valid if the dest is coded as an asterisk (*) for a blackhole filter.

IPv6_IGNORE_RIP_NEIGHBOR statement

Use the IPv6_IGNORE_RIP_NEIGHBOR statement to specify that IPv6 RIP routing table messages from the specified gateway are to be ignored. This option serves as an IPv6 RIP input filter.

Syntax

```
►►──IPv6_Ignore_Rip_Neighbor──IP_address=link_local_address──────────────────►◄
```

Parameters

IP_address
Specifies the link-local IP address, in colon-hexadecimal format, of the gateway from which routing table messages are ignored. For multiple IP addresses, the statement must be repeated for each IP address.

IPv6_ORIGINATE_RIP_DEFAULT statement

Indicates under what conditions IPv6 RIP supports Default route (destination/prefix_len ::/0) generation.

Guideline: This statement determines whether or not a default route is considered available by OMPROUTE IPv6 RIP. The SEND_DEFAULT_ROUTES parameter on the IPV6_RIP_INTERFACE statement determines whether or not an available default route is advertised by a particular IPV6 RIP interface.

Syntax

```
                              ┌─Condition=Always──────┐  ┌─Cost=1──────┐
►►──IPv6_Originate_RIP_Default─┼───────────────────────┼──┼─────────────┼───────►
                              └─Condition──=──condition─┘  └─Cost──=──cost─┘

  ┌─Accept_Default=NO──────┐
►─┼────────────────────────┼──────────────────────────────────────────────────►◄
  └─Accept_Default──=──value─┘
```

Parameters

Condition
Condition for when IPv6 RIP is to advertise this router as a default router. Valid values are:

Always
Always originate IPv6 RIP default. This is the default value.

Never Never advertise this router as a default IPv6 RIP router.

OSPF Advertise this router as a default IPv6 RIP router if there are any IPv6 OSPF routes available.

Cost
Specifies the cost that IPv6 RIP dvertises with the default route that it originates. Valid values are in the range 1 - 16. The default value is 1.

Accept_Default
Specifies whether or not OMPROUTE IPV6 RIP accepts default routes from inbound IPv6 RIP packets whose cost is higher than default routes originated by the host.

Tip: OMPROUTE IPv6 RIP always accepts default routes from inbound IPv6 RIP packets whose cost is lower than default routes originated by the host.

A value of YES indicates that OMPROUTE IPv6 RIP replaces this router's originated default route with a default route learned from inbound IPv6 RIP packets, even if that learned default route has a higher cost than this router's originated default route.

Result: When YES is coded, this router's originated default route is only used if no other default routes are learned from inbound IPv6 RIP packets.

A value of NO indicates that OMPROUTE IPv6 RIP replaces this router's originated default route with a default route learned from inbound IPv6 RIP packets only when the learned IPv6 RIP route has a lower cost than this router's originated default route. This is the default value.

Result: When this parameter value is **NO** (either coded or by default), and the other parameters in this statement take their default values (CONDITION=ALWAYS and COST=1), OMPROUTE IPv6 RIP never accepts default routes learned from IPv6 RIP packets because it is not possible to learn an IPv6 RIP route whose cost is less than 1.

IPv6_RIP_INTERFACE statement

Use the IPv6_RIP_INTERFACE statement to configure the IPv6 RIP parameters for each IP interface. Replicate this statement in the configuration file for each IP interface over which IPv6 RIP operates.

Syntax

```
►►──IPv6_RIP_Interface──Name──=──interface_name──────────────────────────►


   ┌──────────────────────────────┐  ┌─Receive_RIP=YES────────┐
►──▼─┬──────────────────────────┬─┴──┼────────────────────────┼───────────►
     └─Prefix──=──ipaddr/prefix_len─┘  └─Receive_RIP──=──value─┘

  ┌─Receive_Prefix_Routes=YES─────────┐  ┌─Receive_Host_Routes=NO──────────┐
►─┼───────────────────────────────────┼──┼─────────────────────────────────┼──►
  └─Receive_Prefix_Routes──=──value─┘  └─Receive_Host_Routes──=──value─┘


   ┌───────────────────────────────┐  ┌─Send_Only=ALL──────────────┐
►──▼─┬───────────────────────────┬─┴──┼────────────────────────────┼──────────►
     └─Filter=(type,dest/prefix_len)─┘  └─Send_Only──=──(values)─┘

  ┌─Send_RIP=YES─────────┐  ┌─Send_Default_Routes=NO─────────┐
►─┼──────────────────────┼──┼────────────────────────────────┼────────────────►
  └─Send_RIP──=──value─┘  └─Send_Default_Routes──=──value─┘


  ┌─Send_Prefix_Routes=YES─────────┐  ┌─Send_Static_Routes=NO──────────┐
►─┼────────────────────────────────┼──┼────────────────────────────────┼──────►
  └─Send_Prefix_Routes──=──value─┘  └─Send_Static_Routes──=──value─┘


  ┌─Send_Host_Routes=NO──────────┐
►─┼──────────────────────────────┼────────────────────────────────────────────►
  └─Send_Host_Routes──=──value─┘
```

```
   ┌─Send_Router_Advertisement_Routes=YES────┐
►──┼─────────────────────────────────────────┼──────────────────────────────────────►
   └─Send_Router_Advertisement_Routes──=──value─┘

   ┌─Send_Poisoned_Reverse_Routes=YES────┐  ┌─In_Metric=1──────────┐
►──┼─────────────────────────────────────┼──┼──────────────────────┼───────────────►
   └─Send_Poisoned_Reverse_Routes──=──value─┘  └─In_Metric──=──metric─┘

   ┌─Out_Metric=0─────────┐  ┌─Max_Xmit_Time=120──────┐
►──┼──────────────────────┼──┼────────────────────────┼───────────────────────────►
   └─Out_Metric──=──metric─┘  └─Max_Xmit_Time──=──time─┘

   ┌─Min_Xmit_Time=0.5──────┐  ┌─RT_Gain=0.125──────┐
►──┼────────────────────────┼──┼────────────────────┼─────────────────────────────►
   └─Min_Xmit_Time──=──time─┘  └─RT_Gain──=──value──┘

   ┌─Variance_Gain=0.25──────┐  ┌─Variance_Mult=2──────┐
►──┼─────────────────────────┼──┼──────────────────────┼──────────────────────────►
   └─Variance_Gain──=──value─┘  └─Variance_Mult──=──mult─┘

   ┌─Delay_Acks=NO──────┐
►──┼────────────────────┼──────────────────────────────────────────────────────────►◄
   └─Delay_Acks──=──value─┘
```

Parameters

Name

The name of the interface. This name must match the interface name coded on the INTERFACE statement in the TCP/IP profile. Valid values are any character string of 1 - 16 characters in length. Wildcard names (terminating in *) can be coded. For example, OSAQDIO* would match stack interfaces named OSAQDIO1, OSAQDIO2, OSAQDIOABC, and so on.

Tips:

- For more information about how wildcard interfaces are parsed, see the step about defining IPv6 interfaces in z/OS Communications Server: IP Configuration Guide.
- For the names to use when defining IPv6 dynamic XCF interfaces, see the routing information in z/OS Communications Server: IP Configuration Guide.

Prefix

Specifies a prefix that is on the link to which the interface attaches. For each configured Prefix parameter, OMPROUTE adds a direct route to the prefix identified by the first *prefixlen* bits of *ipaddr*. Valid values for *ipaddr* are any valid colon-hexadecimal IPv6 address. Valid values for *prefixlen* are in the range 1 - 127. The prefix identified by the first *prefixlen* bits of *ipaddr* must not be a multicast prefix, a link-local prefix, or all zeros.

Guideline: If IPv6 Router Discovery is in use by the routers on the link, prefixes being advertised as on-link by the routers by way of Router Discovery should not be configured using this keyword. However, if IPv6 Router Discovery is not in use by the routers on the link or there is a need to supplement the list of prefixes being advertised as on-link by the routers, this keyword can be used. If the same prefix is configured using this keyword and learned from Router Discovery, the route in the TCPIP stack's route table is the route added by OMPROUTE as a result of this keyword being specified. Any route for the same prefix that is learned from Router Discovery is ignored as long as the OMPROUTE route exists.

`Receive_RIP`
Specifies whether IPv6 RIP updates are accepted over this interface. Valid values are:

YES IPv6 RIP packets are received over this interface, subject to other filters. This is the default value.

NO No IPv6 RIP packets are received over this interface, regardless of any other filters.

`Receive_Prefix_Routes`
Specifies whether or not to learn routes for prefixes over this interface. If this is not set, only prefixes explicitly allowed using the IPv6_Accept_RIP_Route configuration statement are accepted on this interface. Valid values are YES or NO.

`Receive_Host_Routes`
Specifies whether or not to learn routes for hosts over this interface. If this is not set, only hosts explicitly allowed using the IPv6_Accept_RIP_Route configuration statement are accepted on this interface. Valid values are YES or NO.

`filter`
Multiple filter parameters can be coded on a IPv6_RIP_Interface statement. When specified on the IPv6_RIP_Interface statement, the filter parameter applies only to the corresponding IPv6 RIP interface. The IPv6_RIP_Filter statement can also be coded stand-alone in the OMPROUTE configuration file (nosend and noreceive only) to apply to all configured IPv6 RIP interfaces.

The type can be any of the following values:

Value	Description
nosend	Specifies that routes matching the *dest* and *prefix_len* are not to be sent over this interface. This option serves as an IPv6 RIP output filter.
noreceive	Specifies that routes matching the *dest* and *prefix_len* are to be ignored in messages received over this interface. This option serves as an IPv6 RIP input filter.
send	Specifies that routes matching the *dest* and *prefix_len* are to be sent over only this interface (or any other IPv6 RIP interface with an equivalent filter). This option serves as an IPv6 RIP output filter and can be used for inbound and outbound traffic splitting.
send_cond	Specifies that routes matching the *dest* and *prefix_len* are to be sent over only this interface when this interface is active (or any other active IPv6 RIP interface with an equivalent filter). If this interface is inactive, the routes can be sent over other interfaces. This option serves as an IPv6 RIP output filter and can be used for inbound and outbound traffic splitting.

receive
Specifies that routes matching the *dest* and *prefix_len* are to be received over only this interface (or any other IPv6 RIP interface with an equivalent filter). If received over other IPv6 RIP interfaces, the routes are discarded. This option serves as an IPv6 RIP input filter.

receive_cond
Specifies that routes matching the *dest* and *prefix_len* are to be received over only this interface when this interface is active (or any other active IPv6 RIP interface with an equivalent filter). If this interface is inactive, the routes can be received over all other active IPv6 RIP interfaces. This option serves as an IPv6 RIP input filter.

The dest specifies the destination route in colon-hexadecimal format. Alternatively, an asterisk (*) can be coded in conjunction with the nosend and noreceive filter types. This serves as a blackhole filter that can be used to filter out all routes sent or received over an interface. This should be used in conjunction with either additional send or receive filters to allow only certain routes to be received, or advertised over an interface or set of interfaces.

Tip: If the blackhole nosend filter is used, it does not filter out the sending of the default route when the Originate_RIP_Default statement is also configured.

The *prefix_len* specifies the number of significant bits in the destination to be filtered. If not coded, the default *prefix_len* is 128, meaning apply the filter to the dest route as coded. Coding the *prefix_len* has no meaning and is not valid if the dest is coded as an asterisk (*) for a blackhole filter.

`Send_Only`
Specifies send restrictions. Multiple values can be coded by separating the values with commas, unless ALL is coded. The valid values are:

ALL Specifies no send restrictions.

VIRTUAL
Sends virtual IP addresses.

DEFAULT
Sends the default route.

DIRECT
Sends direct routes.

TRIGGERED
Only sends routes when requested or when a route becomes inactive (metric 16).

VIRTUAL, DEFAULT, and DIRECT are OR'd together to determine what should be sent. Thus, coding SEND_ONLY=(VIRTUAL, DEFAULT) sends virtual IP addresses and the default route. When ALL is coded, it must not be enclosed within parentheses. When any of the other possible values are coded, they must be enclosed within parentheses.

When specified on the IPv6_RIP_Interface statement, the Send_Only parameter applies only to the corresponding IPv6 RIP interface. The IPv6_RIP_Send_Only statement can also be coded stand-alone in the OMPROUTE configuration file to apply to all IPv6 RIP interfaces.

`Send_RIP`
Determines whether or not IPv6 RIP advertisements are sent over this interface. Valid values are YES or NO.

Send_Default_Routes
Advertise the default route (destination/prefix_len ::/0), if it is available, in IPv6 RIP responses sent from this IP source address. Valid values are YES or NO. If DEFAULT is coded on the Send_Only parameter or the stand-alone IPv6_RIP_Send_Only statement, the Send_Default_Routes parameter is ignored and is set to YES.

Send_Prefix_Routes
Advertise all prefix routes in IPv6 RIP responses sent from this IP address. Valid values are YES or NO.

Send_Static_Routes
Advertise static and direct routes in IPv6 RIP responses sent from this IP source address. Split horizon is applied; that is, static routes configured over an interface are not included in IPv6 RIP responses sent from that interface. Valid values are YES or NO.

Send_Host_Routes
Advertise host routes in IPv6 RIP responses sent from this IP source address. In this context, a host route is one with a prefix length of 128. Valid values are YES or NO.

Send_Router_Advertisement_Routes
Advertise router advertisement routes in IPv6 RIP responses sent from this IP source address. These are routes that have been learned by the stack using IPv6 Router Discovery and that OMPROUTE has learned from the stack. Split horizon is applied; that is, router advertisement routes learned over an interface are not included in IPv6 RIP responses sent from that interface. Valid values are YES or NO.

Send_Poisoned_Reverse_Routes
Advertise poisoned reverse routes over the interface corresponding to the next hop. A poison reverse route is one with an infinite metric (16). Valid values are YES or NO. If NO is specified, OMPROUTE still uses split horizon.

In_Metric
Specifies the value of the metric to be added to IPv6 RIP routes received over this interface prior to installation in the routing table. Valid values are in the range 1 - 15.

Out_Metric
Specifies the value of the metric to be added to IPv6 RIP routes advertised over this interface. Valid values are in the range 0 - 15.

Retransmit Parameters

The following parameters are used by OMPROUTE to set values in the routes added to the TCP/IP route table which use this interface. The values affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a certain number of retransmits, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times packets are retransmitted before the connection times out differs for initial connection establishment and for data packets. For initial connection establishment, the initial time interval is set at approximately 3 seconds, and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval and are retransmitted 15 times before the connection is timed out. All of the following parameters affect the data packet retransmission algorithm. Only the Min_Xmit_Time parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

Max_Xmit_Time
Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to time out. Specifying Max_Xmit_Time assures that the interval time never exceeds the specified limit. The minimum value that can be specified for Max_Xmit_Time is 0. The maximum is 999.990. The default is 120 seconds. This parameter affects the initial connection establishment retransmission timeout for all APIs, except the Pascal API (TcpOpen), that are using the socket connect function.

Min_Xmit_Time
Sets a minimum retransmit interval. Increasing this value might increase the amount of time it takes for TCP to time out a connection. The minimum value that can be specified for Min_Xmit_Time is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

RT_Gain
This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet's RTT has on the average. The minimum value that can be specified for RT_Gain is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

Variance_Gain
This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum value that can be specified for Variance_Gain is 0. The maximum value is 1.0. The default is 0.25 . This parameter does not affect initial connection retransmission.

Variance_Mult
This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT has on calculating the retransmission interval. The minimum value that can be specified for Variance_Mult is 0. The maximum value is 99.990. The default is 2. This parameter does not affect initial connection retransmission.

Delay_Acks
The delay acknowledgments value that is added to the routing tables for routes that use this interface. Specify YES to delay transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specify NO to return acknowledgments immediately when a packet is received with the PUSH bit on in the TCP header. This parameter affects only connections that use the routes associated with this interface.

Even if you specify YES, you can override the delay acknowledgments behavior can be overridden by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements. A value of NO can override the specification of the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

Valid values are YES and NO. The default value is YES.

IPv6_RIP_SEND_ONLY statement

Use the IPv6_RIP_SEND_ONLY statement to allow for the specification of the types of routes that are to be included in advertisements sent over IPv6 RIP interfaces. The IPv6_RIP_Send_Only statement can be coded stand-alone in the OMPROUTE configuration file to apply to all IPv6 RIP interfaces.

Syntax

```
     ┌─IPv6_RIP_Send_Only=ALL──────────┐
►►───┼─────────────────────────────────┼──────────────────────────────────────►◄
     └─IPv6_RIP_Send_Only──=──(values)─┘
```

Parameters

(values)
Specifies send restrictions. Multiple values can be coded by separating the values with commas, unless ALL is coded. The valid values are:

ALL Specifies no send restrictions.

VIRTUAL
Sends virtual IP addresses.

DEFAULT
Sends the default route.

DIRECT
Sends direct routes.

TRIGGERED
Only sends routes when requested or when a route becomes inactive (metric 16).

VIRTUAL, DEFAULT, and DIRECT are OR'd together to determine what should be sent. Thus, coding SEND_ONLY=(VIRTUAL, DEFAULT) sends virtual IP addresses and the default route. When ALL is coded, it must not be enclosed within parentheses. When any of the other possible values are coded, they must be enclosed within parentheses.

When specified on the IPv6_RIP_Send_Only statement in the OMPROUTE configuration file, this statement applies to all IPv6 RIP interfaces. The SEND_ONLY parameter can also be coded on the IPv6_RIP_Interface statement. When specified on the IPv6_RIP_Interface statement, the SEND_ONLY parameter applies only to the corresponding IPv6 RIP interface.

Common configuration statements for RIP and OSPF

This topic contains descriptions of the common configuration statements:

- DEFAULT_ROUTE
- ROUTESA_CONFIG
- INTERFACE
- GLOBAL_OPTIONS

- IPV6_DEFAULT_ROUTE
- IPV6_INTERFACE

DEFAULT_ROUTE statement

Use the DEFAULT_ROUTE statement to specify IPv4 default routes to OMPROUTE. Default routes are created in the following ways:

- Specify a BEGINROUTES or GATEWAY statement in the TCP/IP profile for a default route in the main route table
- Specify a Policy Agent RouteTable statement for a default route in a policy-based route table
- Specify a Default_Route statement
- Let the default route be learned by routing protocol

You can configure up to 16 default routes using the DEFAULT_ROUTE statement. Each default route is added to the main route table. A default route is also added to any policy-based route tables if the interface and next-hop values associated with the default route are compatible with the dynamic routing parameters defined for those route tables. The Send_Default_Routes keyword on the RIP_Interface statement indicates whether the default routes over that interface should be advertised.

Syntax

```
                 ┌─────────────────────────────────────────────────────┐
                 ▼                                                     │
►►──Default_Route───Name──=──interface_name ── Next_Hop──=──ip_address─┴──────────►◄
```

Parameters

Name
The name of the interface used in the default route. This name must match a link name coded on the HOME statement or the interface name coded on an IPv4 INTERFACE statement in the TCP/IP profile. The name can be any 16 characters.

Restriction: VIPA interfaces cannot be used for the *interface_name* value.

Next_Hop
IP address of the next hop used in the default route.

ROUTESA_CONFIG statement

Use the ROUTESA_CONFIG statement to configure the OMPROUTE OSPF subagent.

Syntax

```
                  ┌─Community="public"─────────┐
►►──ROUTESA_Config─┼────────────────────────────┼─────────────────────────────►
                  └─Community=──community_string─┘
```

```
   ┌─Agent=161──────────────────┐  ┌─Enabled=YES──┐
►──┼────────────────────────────┼──┼──────────────┼──────────────────────────►◄
   └─Agent=──agent_port_number──┘  └─Enabled=──value─┘
```

Parameters

Community

A character string of 1 - 32 characters enclosed in double quotation marks ("") used as the community name (or password) in establishing contact with the SNMP agent. For the OMPROUTE subagent to communicate with the z/OS Communications Server SNMP agent, the community name specified on the COMMUNITY keyword must match one that is defined in the PW.SRC or SNMPD.CONF data set configured to the SNMP agent or the -c parameter when the SNMP agent is started.

For more information about how the community name is used to permit access to the SNMP agent, see Step 1: Configure the SNMP agent (OSNMPD), in z/OS Communications Server: IP Configuration Guide. The default value is public.

Tip: The community name is case sensitive.

Agent

A port number in the range 1 - 65 535 used in establishing communication with the SNMP agent. For the OMPROUTE subagent to communicate with the z/OS Communications Server SNMP agent, the port number specified must match the port number specified on the -p parameter when the SNMP agent is started. The default value is 161.

Enabled

A value of YES indicates that the OMPROUTE subagent should be started during OMPROUTE initialization. If there are no active OSPF interfaces, the OMPROUTE subagent returns *noSuchInstance* for all GET and GETNEXT requests. By default, the OMPROUTE subagent is started when OMPROUTE is started.

A value of NO indicates that the OMPROUTE subagent should not be started. Specify this keyword if little or no OSPF SNMP data is requested from this OSPF image. SNMP MIB objects supported by the TCP/IP SNMP agent and TCP/IP subagent (other than the OMPROUTE subagent) are still available. For information about which MIB objects are supported by the SNMP agent and OMPROUTE subagent, see the z/OS Communications Server: IP User's Guide and Commands.

Examples

```
ROUTESA_CONFIG COMMUNITY="USACCESS" AGENT=528
ROUTESA_CONFIG ENABLED=NO
```

Usage notes

- If ENABLED=NO is specified, the OMPROUTE subagent is *not* started during OMPROUTE initialization. If the ROUTESA_CONFIG statement itself is *not* specified, the OMPROUTE subagent is started (this is the default).
- The community string is case sensitive and must be 1 - 32 characters. It is not converted to uppercase by profile processing.
- A MODIFY command can be used to start or stop the OMPROUTE subagent, but the setting of the parameters cannot be changed unless OMPROUTE is recycled.

INTERFACE statement

Use the INTERFACE statement to allow certain values to be specified for generic IPv4 interfaces, which are interfaces that are neither OSPF nor RIP interfaces. Each IPv4 interface that is neither an OSPF nor an RIP interface should be configured to OMPROUTE using the INTERFACE statement unless it is a non-point-to-point interface and the default values for Subnet_Mask and MTU are acceptable for that interface.

Tip: To display information about INTERFACEs, use the **d tcpip,tcpname,** OMP,GENERIC commands.

Syntax

```
►►──Interface──IP_address──=──ip_address──Name──=──interface_name──────────────────►

                                                       ┌─MTU=576────────┐
►─Subnet_Mask──=──mask──┬───────────────────────────┬──┼────────────────┼──────────►
                        └─Destination_Addr──=──address─┘  └─MTU──=──size─┘

  ┌─Max_Xmit_Time=120────┐   ┌─Min_Xmit_Time=0.5────┐
►─┼──────────────────────┼───┼──────────────────────┼───────────────────────────────►
  └─Max_Xmit_Time──=──time─┘ └─Min_Xmit_Time──=──time─┘

  ┌─RT_Gain=0.125──────┐  ┌─Variance_Gain=0.25─────┐
►─┼────────────────────┼──┼────────────────────────┼────────────────────────────────►
  └─RT_Gain──=──value──┘  └─Variance_Gain──=──value─┘

  ┌─Variance_Mult=2──────┐   ┌─Delay_Acks=YES───────┐
►─┼──────────────────────┼───┼──────────────────────┼──────────────────────────────►◄
  └─Variance_Mult──=──mult─┘ └─Delay_Acks──=──value─┘
```

Parameters

IP_address

The IP address can be a valid IP address that is configured on the system or it can be specified with asterisks (*) as wildcards. The valid wildcard specifications are below. The result of coding a wildcard value is that all configured interfaces whose IP address matches the wildcard are configured as interfaces. Configured interface IP addresses and names are matched against possible wildcards in the order they appear below with the name and any matching wildcard being the best match, x.y.z.* being second best, and so forth.

```
interface name and any matching wildcard
x.y.z.*
x.y.*.*
x.*.*.*
*.*.*.* - Same as ALL
ALL - Same as *.*.*.*
```

Tip: For more information about how wildcard interfaces are parsed, see this Method of assigning interface definitions to stack interfaces (wildcard and explicit): in z/OS Communications Server: IP Configuration Guide.

Because a stack could have a large number of Dynamic VIPAs (DVIPAs) defined, as well as DVIPA ranges, additional wildcard capabilities exist on the INTERFACE statement for use only with DVIPAs. Ranges of DVIPA interfaces can be defined using the subnet mask parameter on the INTERFACE statement. The range defined in this way is all the IP addresses that fall within the subnet defined by the mask and the IP address.

When this type of wildcarding is being used, the value of the IP_ADDRESS parameter must be the subnet number of the range. For example, the following code defines a range of six addresses (9.67.101.9 to 9.67.101.14) that can be used for DVIPA addresses and matches any DVIPA interface that fall into the 9.67.101.8/29 subnet:

```
IP_ADDRESS= 9.67.101.8
SUBNET_MASK= 255.255.255.248
```

Alternatively, the following code is not because 9.67.101.17 is an address within the subnet range, not the subnet number itself (that would be 9.67.101.16). This second definition only matches an interface whose home address is 9.67.101.17.

```
IP_ADDRESS= 9.67.101.17
SUBNET_MASK=255.255.255.248
```

Name
The name of the interface. A valid value is any string 1 - 16 characters in length.

Rules:

- If this is not a wildcard interface definition, the name must match the link name that is coded for the corresponding IP address on the HOME statement or the interface name coded for the corresponding IPv4 INTERFACE statement in the TCP/IP profile.
- If this is a wildcard interface definition, then this parameter is used in conjunction with the defined wildcard IP address when searching for definitions to match a stack interface. For more details about this process, see method of assigning interface definitions to stack interfaces (wildcard and explicit): in z/OS Communications Server: IP Configuration Guide.

For Dynamic VIPA (DVIPA), link names are assigned programmatically by the stack when the DVIPA is created; therefore, the name field set on the INTERFACE statement is ignored by OMPROUTE for DVIPAs.

Subnet_Mask
Subnet mask for the associated interface's IP address. If you configure this interface in the TCP/IP profile using the IPv4 INTERFACE statement and you configure a subnet mask on that statement that does not match the value that you specify on this parameter, OMPROUTE issues message EZZ8164I and uses this subnet mask.

Destination_Addr
IP address of the host at the remote end of this interface. This parameter is valid only for point-to-point links. If this parameter is not specified for a point-to-point link, a route to the host at the remote end of the interface is not added to the appropriate TCP/IP route tables (main and policy-based tables). A subnet route for the interface is added when OMPROUTE is initialized whether or not this parameter is specified.

MTU
The maximum transmission unit size that OMPROUTE adds to the appropriate routing tables (main and policy-based tables) for routes that use this interface. Valid values are in the range 0 - 65535. If you configure this interface in the TCP/IP profile using the IPv4 INTERFACE statement and you configure an MTU on that statement and the MTU that you configure on that statement

does not match the MTU (the configured value or the default value) on this statement, OMPROUTE issues message EZZ8163I and uses the MTU value on this statement.

Tip: See z/OS Communications Server: IP Configuration Guide, in section Maximum transmission unit considerations, for additional information about how TCP/IP uses the MTU to determine the largest size frame to send.

Retransmit Parameters

The following parameters are used by OMPROUTE to set values in the routes which use this interface that are added to the TCP/IP route tables. The values affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a certain number of retransmits, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times packets are retransmitted before the connection times out differs for initial connection establishment and for data packets . For initial connection establishment, the initial time interval is set at approximately 3 seconds, and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval and are retransmitted 15 times before the connection is timed out. All of the following parameters affect the data packet retransmission algorithm. Only the Min_Xmit_Time parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

Max_Xmit_Time
Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to time out. Specifying Max_Xmit_Time assures that the interval time never exceeds the specified limit. The minimum value that can be specified for Max_Xmit_Time is 0. The maximum is 999.990. The default is 120 seconds. This parameter affects the initial connection establishment retransmission timeout for all APIs, except the Pascal API (TcpOpen), that are using the socket connect function.

Min_Xmit_Time
Sets a minimum retransmit interval. Increasing this value might increase the amount of time it takes for TCP to time out a connection. The minimum value that can be specified for Min_Xmit_Time is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

RT_Gain
This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet's RTT has on the average. The minimum value that can be specified for RT_Gain is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

Variance_Gain
This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum value that can be specified for Variance_Gain is 0. The maximum value is 1.0. The default is 0.25. This parameter does not affect initial connection retransmission.

Variance_Mult
This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT has on calculating the retransmission interval. The minimum value that can be specified for Variance_Mult is 0. The maximum value is 99.990. The default is 2. This parameter does not affect initial connection retransmission.

Delay_Acks
The delay acknowledgments value that is added to the routing tables for routes that use this interface. Specify YES to delay transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specify NO to return acknowledgments immediately when a packet is received with the PUSH bit on in the TCP header. This parameter affects only connections that use the routes associated with this interface.

Even if you specify YES, you can override the delay acknowledgments behavior can be overridden by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements. A value of NO can override the specification of the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

Valid values are YES and NO. The default value is YES.

GLOBAL_OPTIONS statement

Use the GLOBAL_OPTIONS statement to configure miscellaneous options to OMPROUTE which apply to OSPF, RIP, or neither.

Syntax

```
                      ┌─Ignore_Undefined_Interfaces=NO────┐
►►──Global_Options────┼───────────────────────────────────┼──────────────────────────►◄
                      └─Ignore_Undefined_Interfaces=value─┘
```

Parameters

Ignore_Undefined_Interfaces
Instructs OMPROUTE on how to handle stack interfaces that are not configured by way of OSPF_INTERFACE, RIP_INTERFACE, IPV6_RIP_INTERFACE, IPV6_OSPF_INTERFACE, INTERFACE, or IPV6_INTERFACE statements, either explicit or wildcard. NO indicates that OMPROUTE configures such interfaces with default values (including setting the MTU size to 576 and using the class mask as the subnet mask and overriding stack definition values with these default values for IPv4 interfaces), and possibly advertises these interfaces to the rest of the network if OSPF settings or RIP filters permit it.

Result: By definition, the network class masks that are used for undefined IPv4 interfaces if GLOBAL_OPTIONS Ignore_Undefined_Interfaces=NO is coded, or the GLOBAL_OPTIONS statement is not coded, (thereby taking the default of NO) are as follows:

- Class A: 255.0.0.0
- Class B: 255.255.0.0
- Class C: 255.255.255.0

A YES value indicates that OMPROUTE ignores these interfaces, does not configure them, and does not advertise them under any circumstances. IF YES is coded, OMPROUTE does not advertise these interfaces or their attached subnets or prefixes, and it does not update any stack definition values. Static routes coded in the TCP/IP profile over interfaces that are ignored by OMPROUTE are still advertised by OMPROUTE into OSPF, RIP, or both if appropriate filters and settings permit the advertisement of static routes.

Guideline: Code YES to get additional reconfiguration capability. You can use OMPROUTE reconfiguration to add a definition for an interface that has been defined to the stack but is ignored by OMPROUTE. However, OMPROUTE does not associate the interface with the new definition until it has been deleted from the stack and re-added.

IPv6_DEFAULT_ROUTE statement

Use the IPv6_DEFAULT_ROUTE statement to allow IPv6 default routes to be specified to OMPROUTE. IPv6 default routes are created using any of the following methods:

- BEGINROUTES statement
- IPv6_Default_Route statement
- Learned by routing protocol
- Router advertisements

When IPv6 default routes are specified using more than one of these methods, the method that is used to create the default routes is determined according to the following list, in order of descending precedence:

1. Non-replaceable static default routes specified using the BEGINROUTES statement
2. Default routes learned by routing protocol
3. Default routes specified using the IPv6_Default_Route statement
4. Router advertisements
5. Replaceable static default routes specified using the BEGINROUTES statement

Up to 16 default routes can be configured using this IPv6_Default_Route statement. The Send_Default_Routes keyword on the IPv6_RIP_Interface statement indicates whether or not to advertise the IPv6 default routes over that interface.

Syntax

```
                     ┌──────────────────────────────────────────────────┐
                     ▼                                                  │
►►──IPv6_Default_Route───Name = interface_name  Next_Hop = ip_address──┴──────────►◄
```

Parameters

Name

The name of the interface used in the default route. This name must match an interface name coded on the INTERFACE statement in the TCP/IP profile. Valid values are any 16 characters.

Restriction: You cannot use VIPA interfaces for the *interface_name* value.

Next_Hop

IP address of the next hop used in the default route, in colon-hexadecimal format. This IP address must be reachable using a direct route over the specified interface. If it is not, this next hop is not installed.

IPv6_INTERFACE statement

Use the IPv6_INTERFACE statement to allow certain values to be specified for generic IPv6 interfaces, which are interfaces that are neither IPv6 OSPF nor IPv6 RIP interfaces. If GLOBAL_OPTIONS is coded with IGNORE_UNDEFINED_INTERFACES=YES, then IPv6 interfaces that are not used for routing but that OMPROUTE should be aware of should be coded in the OMPROUTE configuration file. If that option is not coded, it is not necessary to code all non-routing IPv6 interfaces to OMPROUTE if default values are acceptable and you do not need to code additional prefixes on the interface.

Tip: Use the d tcpip, tcpname, omproute, and generic6 commands to display information about IPv6_INTERFACEs.

Syntax

```
                                        ┌───────────────────────────┐
                                        ▼                           │
►►──IPv6_Interface──Name=interface_name────┬─────────────────────────┬┴───────────────►
                                          └─Prefix=ipaddr/prefixlen─┘

   ┌─Max_Xmit_Time=120────┐   ┌─Min_Xmit_Time=0.5────┐   ┌─RT_Gain=0.125───┐
►──┼──────────────────────┼───┼──────────────────────┼───┼─────────────────┼────────►
   └─Max_Xmit_Time = time─┘   └─Min_Xmit_Time = time─┘   └─RT_Gain = value─┘

   ┌─Variance_Gain=0.25────┐   ┌─Variance_Mult=2──────┐   ┌─Delay_Acks=YES────┐
►──┼───────────────────────┼───┼──────────────────────┼───┼───────────────────┼────►◄
   └─Variance_Gain = value─┘   └─Variance_Mult = mult─┘   └─Delay_Acks = value─┘
```

Parameters

Name

The name of the interface. This name must match the interface name coded on the INTERFACE statement in the TCP/IP profile. Valid values are any character string of 1 - 16 characters in length. Wildcard names (terminating in *) can be coded. For example, OSAQDIO* would match stack interfaces named OSAQDIO1, OSAQDIO2, OSAQDIOABC, and so on.

Tip: For more information about how wildcard interfaces are parsed, see the step about defining IPv6 interfaces in z/OS Communications Server: IP Configuration Guide.

Prefix
Specifies a prefix that is on the link to which the interface attaches. For each configured Prefix parameter, OMPROUTE adds a direct route to the prefix identified by the first *prefixlen* bits of *ipaddr*. Valid values for *ipaddr* are any valid colon-hexadecimal IPv6 address. Valid values for *prefixlen* are in the range 1 - 127. The prefix identified by the first *prefixlen* bits of *ipaddr* must not be a multicast prefix, a link-local prefix, or all zeros.

Guideline: If IPv6 Router Discovery is in use by the routers on the link, prefixes being advertised as on-link by the routers by way of Router Discovery should not be configured using this keyword. However, if IPv6 Router Discovery is not in use by the routers on the link or there is a need to supplement the list of prefixes being advertised as on-link by the routers, this keyword can be used. If the same prefix is configured using this keyword and learned from Router Discovery, the route in the TCPIP stack's route table is the route added by OMPROUTE as a result of this keyword being specified. Any route for the same prefix that is learned from Router Discovery is ignored as long as the OMPROUTE route exists.

Retransmit Parameters

The following parameters are used by OMPROUTE to set values in the routes added to the TCP/IP route table which use this interface. The values affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a certain number of retransmits, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times packets are retransmitted before the connection times out differs for initial connection establishment and for data packets . For initial connection establishment, the initial time interval is set at approximately 3 seconds, and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval and are retransmitted 15 times before the connection is timed out. All of the following parameters affect the data packet retransmission algorithm. Only the Min_Xmit_Time parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

Max_Xmit_Time
Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to time out. Specifying Max_Xmit_Time assures that the interval time never exceeds the specified limit. The minimum value that can be specified for Max_Xmit_Time is 0. The maximum is 999.990. The default is 120 seconds. This parameter affects the initial connection establishment retransmission timeout for all APIs, except the Pascal API (TcpOpen), that are using the socket connect function.

Min_Xmit_Time
Sets a minimum retransmit interval. Increasing this value might increase the

amount of time it takes for TCP to time out a connection. The minimum value that can be specified for Min_Xmit_Time is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

RT_Gain
This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet's RTT has on the average. The minimum value that can be specified for RT_Gain is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

Variance_Gain
This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum value that can be specified for Variance_Gain is 0. The maximum value is 1.0. The default is 0.25 . This parameter does not affect initial connection retransmission.

Variance_Mult
This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT has on calculating the retransmission interval. The minimum value that can be specified for Variance_Mult is 0. The maximum value is 99.990. The default is 2. This parameter does not affect initial connection retransmission.

Delay_Acks
The delay acknowledgments value that is added to the routing tables for routes that use this interface. Specify YES to delay transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. Specify NO to return acknowledgments immediately when a packet is received with the PUSH bit on in the TCP header. This parameter affects only connections that use the routes associated with this interface.

Even if you specify YES, you can override the delay acknowledgments behavior can be overridden by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements. A value of NO can override the specification of the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

Valid values are YES and NO. The default value is YES.

Interfaces supported by OMPROUTE

Table 25, Table 26 on page 550, and Table 27 on page 551 show the types of interfaces supported by OMPROUTE.

Table 25. Types of IPv4 interfaces (using DEVICE and LINK statements) supported by OMPROUTE

Interface type	Link type	Connectivity	Multi-access broadcast	Non-broadcast multiaccess (NBMA)	Point-to-point	Point-to-multi-point	Futile neighbor state loop detection support
ATM	ATM	ATM network through OSA-2 or OSA-Express in ATM native mode	No	Yes	No	No	No

Table 25. Types of IPv4 interfaces (using DEVICE and LINK statements) supported by OMPROUTE (continued)

Interface type	Link type	Connectivity	Multi-access broadcast	Non-broadcast multiaccess (NBMA)	Point-to-point	Point-to-multi-point	Futile neighbor state loop detection support
CDLC	CDLC	3745/3746 network through NCP	No	No	Yes	No	No
CLAW	IP	pSeries or OEM	No	No	Yes	Yes and no (See Note 2)	No
CTC	CTC	z/OS through channel-channel adapter	No	No	Yes	No	No
HCH	HCH	Another host through a hyperchannel adapter	No	Yes	No	No	No
LCS	IBMTR, ETHERNET, 802.3	LAN through OSA in LCS mode (including ATM LAN emulation), 3172, 2216, Token Ring, Ethernet, Ethernet 802.3, FDDI	Yes	No	No	No	Yes
MPCIPA	IPAQENET, IPAQTR	LAN through OSA-Express in QDIO mode (Gigabit Ethernet, Fast Ethernet, ATM Ethernet LANE, High Speed Token Ring)	Yes	No	No	No	Yes
MPCIPA: IPAQIDIO (for internal QDIO)	IPAQIDIO (for internal QDIO)	Another TCP/IP within same CPC	Yes	No	No	No	Yes
MPCOSA	OSAENET, OSAFDD	Fast Ethernet, FDDI	No	Yes	No	No	No
MPCPTP	MPCPTP	z/OS, pSeries, Cisco CIP, CS/NT, or OEM z/OS,	No	No	Yes	Yes	No
MPCPTP	MPCPTP (for XCF)	Another TCP/IP within same z/OS sysplex	No	No	Yes	Yes	No

Table 25. Types of IPv4 interfaces (using DEVICE and LINK statements) supported by OMPROUTE (continued)

Interface type	Link type	Connectivity	Multi-access broadcast	Non-broadcast multiaccess (NBMA)	Point-to-point	Point-to-multi-point	Futile neighbor state loop detection support
MPCPTP (for IUTSAMEH)	MPCPTP (for IUTSAMEH)	Another TCP/IP within same z/OS sysplex	No	No	Yes	Yes	No
SNAIUCV	SAMEHOST	SNA network through a SNALINK LU 0 application on same z/OS	No	No	Yes	No	No
SNALU62	SAMEHOST	SNA network through SNALINK LU6.2 application on same z/OS	No	No	Yes	No	No
X25NPSI	SAMEHOST	X.25 network through X.25 appl on same z/OS	No	Yes	Yes (See Note 3)	No	No

Notes:

1. For more information about the DEVICE and LINK statements for the interfaces, see Figure 1 on page 48.
2. Becomes point-to-multipoint capable when the P2MP parameter option is specified on the LINK statement.
3. In general, SAMEHOST link type is treated as point-to-point. X.25 NPSI established as a SAMEHOST connection is also treated as NBMA because it is a switched virtual circuit (SVC) that appears to VTAM as a switched link. In a X.25 network, SVCs provide multi-access support but they are not broadcast capable.
4. For more information about IPv4 interfaces using the INTERFACE statement, see Table 26

Table 26. Types of IPv4 interfaces (using INTERFACE statement) supported by OMPROUTE

Interface type	Connectivity	Multi-access broadcast	Non- broadcast multiaccess (NBMA)	Point-to-point	Point-to-multi-point	Futile neighbor state loop detection support
IPAQENET	LAN through OSA-Express in QDIO mode (Gigabit Ethernet, Fast Ethernet, ATM Ethernet LANE)	Yes	No	No	No	Yes

Note: For more information about the alternative INTERFACE statements for the interfaces, see “Summary of INTERFACE statements” on page 141.

Table 27. Types of IPv6 interfaces supported by OMPROUTE

Interface type	**Connectivity**	**Multi-access broadcast**	**Non- broadcast multiaccess (NBMA)**	**Point-to-point**	**Point-to-multi-point**	**Futile neighbor state loop detection support**
IPAQENET6	LAN through OSA-Express in QDIO mode (Gigabit Ethernet, Fast Ethernet, ATM Ethernet LANE)	Yes	No	No	No	Yes
IPAQIDIO6 (for internal QDIO)	Another TCP/IP within the same CEC	Yes	No	No	No	Yes
MPCPTP6	z/OS, pSeries, Cisco CIP, CS/NT, or OEM	No	No	Yes	Yes	No
MPCPTP6 (for IUTSAMEH)	Another TCP/IP within same z/OS sysplex	No	No	Yes	Yes	No
MPCPTP6 (for XCF)	Another TCP/IP within same z/OS sysplex	No	No	Yes	Yes	No
Note: For more information about the INTERFACE statements for the interfaces, see "Summary of INTERFACE statements" on page 141.						

Chapter 12. SNALINK

This topic contains the following information:

- “SNALINK cataloged procedure (SNALPROC)”
- “SNALINK parameters”

SNALINK cataloged procedure (SNALPROC)

This topic shows the SNALINK cataloged procedure (SNALPROC).

```
//SNALINK  PROC MODULE=SNALINK,TCP='TCPIP',APPLID='APPLID'
//*
//* z/OS Communications Server
//* SMP/E Distribution Name: EZAEB01U  SEZAINST(SNALPROC)
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5647-A01
//*               (C) Copyright IBM Corp. 1989, 2001
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R2
//*
//SNALINK  EXEC PGM=&MODULE,REGION=2048K,TIME=1440,PARM='&TCP &APPLID'
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*
//*       The SYSMDUMP DD statement will cause MVS to provide
//*       an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*       SYSTCPD explicitly identifies which data set is to be
//*       used to obtain the parameters defined by TCPIP.DATA
//*       when no GLOBALTCPIPDATA statement is configured.
//*       See the IP Configuration Guide for information on
//*       the TCPIP.DATA search order.
//*       The data set can be any sequential data set or a member of
//*       a partitioned data set (PDS).
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Figure 20. SNALINK cataloged procedure (SNALPROC)

SNALINK parameters

The system parameters required by SNALINK are passed by the PARM parameter on the EXEC statement of the SNALINK cataloged procedure. The parameters are order-dependent and appear in the following list:

DEBUG
Enables detailed tracing into an internal buffer. If specified, it must be the first parameter in the list.

TCP=‘tcpid’
Specifies the name of the TCP/IP address space, in quotation marks.

APPLID=‘*applid*’
Specifies the name of the VTAM application (LU name), in quotation marks, that SNALINK uses for this system.

max_ru_size
This parameter is optional, and is the maximum RU size in hexadecimal. The default size is 88 (in form *mn*). Set *max_ru_size* to specify the maximum request or response unit (RU) size that SNALINK sends. The value is of the form *mn*, where *m* is between 8 and F, and *n* is between 0 and F. The corresponding maximum RU size is $m2^n$ (*m* multiplied by 2 to the power of *n*). Use the largest size that works on your SNA network, to provide the best performance and the least overhead. See z/OS Communications Server: SNA Programming for more information about this parameter as well as z/OS Communications Server: IP Configuration Guide.

max_session
The maximum number of sessions; a decimal value from 1 to 9 999. The default value is 6. To use different values for *max_session*, you also have to specify the *max_ru_size*.

retry
The delay time for VTAM to try sense codes again. It has the following format: *hhmm.* Where:

hh	Hours 0 - 24
mm	Minutes 0 - 59

For example:
- 0005 is a 5-minute delay.
- 0200 is a 2-hour delay.
- 1030 is a 10-hour and 30-minute delay

The default delay is 15 minutes and the maximum delay is 24 hours. To use a different retry interval, you must specify both *max_ru_size* and *max_session*.

session_type
Defines the SNALINK communication session mode. The *session_type* can have the values of SINGLE, DUAL, or be omitted. If the parameter is omitted, *session_type* defaults to DUAL. If the *session_type* is set to SINGLE, SNALINK creates a single duplex session. If DUAL is specified, SNALINK creates two sessions, a send session and a receive session. Like *max_session* and *retry*, if *session_type* is specified, you must also specify the previous parameters.

NCPROUTE and 3745 Communication Controller Ethernet links require *session_type* of SINGLE.

Chapter 13. SNALINK LU6.2

This topic contains the following information:
- "SNALINK LU6.2 cataloged procedure (LU62PROC)"
- "Sample SNALINK LU6.2 configuration data set (LU62CFG)" on page 556
- "Summary of SNALINK LU6.2 configuration statements" on page 557
- "SNALINK LU6.2 configuration statements" on page 557

The DD statements in the cataloged procedure should be defined as follows:

DD Name	Description
SYSTCPD	TCPIP.DATA configuration data set
LU62CFG	SNALINK LU6.2 configuration data set
SYSPRINT	Runtime diagnosis or trace output
SYSUDUMP	User abend dump output (optional)

SNALINK LU6.2 cataloged procedure (LU62PROC)

This topic shows the SNALINK LU6.2 cataloged procedure (LU62PROC).

```
//TCPIPL62 PROC MODULE=SNALNK62
//*
//*        z/OS Communications Server
//*        SMP/E Distribution Name: EZAEB023
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5647-A01
//*               (C) Copyright IBM Corp. 1989, 2001
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R2
//*
//SNALNK62 EXEC PGM=&MODULE,TIME=1440,REGION=256K
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*
//*        The SYSMDUMP DD statement will cause MVS to provide
//*        an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA
//*        when no GLOBALTCPIPDATA statement is configured.
//*        See the IP Configuration Guide for information on
//*        the TCPIP.DATA search order.
//*        The data set can be any sequential data set or a member of
//*        a partitioned data set (PDS).
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
//LU62CFG  DD DSN=TCPIP.SEZAINST(LU62CFG),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

Figure 21. SNALINK LU6.2 cataloged procedure (LU62PROC)

Sample SNALINK LU6.2 configuration data set (LU62CFG)

This topic shows a sample of SNALINK LU6.2 configuration data set (LU62CFG).

```
*----------------------------------------------------------------------*
*   Sample configuration file for SNALNK62                             *
*                                                                      *
*   COPYRIGHT = NONE.                                                  *
*                                                                      *
*----------------------------------------------------------------------*
*
*-- VTAM Application definition:
*
*        ApplID    Password
*        --------  --------
VTAM     LU62APPL  QWERTY
*
*-- Link Definition:
*                                    Idle
*        TCP/IP Link Name  LogMode   Time-out
*        ----------------  --------  --------
LINK     LNKLU62           LU62MODE  600
*
*-- Destination address list for this link:
*
*                          *--- REMOTE ----*  Start
*        IP Address        Send LU  Recv LU   Type
*        ---------------   -------- --------  ----
DEST     192.9.207.39      LU6LBK11 LU6LBK12  INIT
DEST     192.9.207.40      LU6LBK13 =
DEST     192.9.207.41      LU6LBK14 =
*
*-- Buffer specifications:
*
*        Datagram  Add Snd  Snd Queue
*        Max Size  Buffers  Limit
*        -----     -----    -----
BUFFERS  32758     10       11
*
*-- Trace level specifications:
*
*      Trace
*      Level   Connection Range
*      ------  ------------------
TRACE  OFF     ALL
```

Figure 22. Sample of LU62CFG

Summary of SNALINK LU6.2 configuration statements

Table 28 shows a summary of SNALINK LU6.2 configuration statements.

Table 28. Summary of SNALINK LU6.2 configuration statements

Statement	Description	See
BUFFERS	Specifies the allocation of buffer pools for IP datagrams	"BUFFERS" on page 558
DEST	Defines an IP-to-LU mapping for a destination node	"DEST" on page 559
LINK	Defines a link between two TCPIP address spaces	"LINK" on page 560
TRACE	Defines the required level of tracing for the connections	"TRACE" on page 561
VTAM	Specifies the VTAM application for the connection	"VTAM" on page 561

SNALINK LU6.2 configuration statements

This topic contains the statements for the SNALINK LU6.2 configuration data set.

Statement syntax

Rules: Use the following syntax rules:

- Each statement must be entered on a separate line in the configuration data set.

- Each statement consists of a keyword, followed by one or more parameter fields separated by a blank.
- Case is not significant and leading blanks are ignored.
- Comment lines are marked with an asterisk (*) in column 1.

Statement ordering

Restrictions: Follow these restrictions for statements:

- At least one of each type of statement, except the TRACE statement, must appear in the configuration data set. TRACE is the only optional statement.
- Only one BUFFERS statement can be defined, and it can appear anywhere in the data set.
- Only one VTAM statement can be defined, and it must appear before the first LINK statement.
- You must have one LINK statement for each network *directly connected* to the SNALINK LU6.2 address space.
- You must have one DEST statement for each distinct IP address on the directly connected networks.
- A DEST statement defining a destination IP address must appear before a TRACE statement for the same destination IP address. The DEST statements for a particular network must appear after the LINK statement for that network and before the next LINK statement.

  For example:

  ```
  LINK
  DEST
  .
  .
  .
  LINK
  ```
- The data set can have any number of TRACE statements. If the ranges specified in any of the TRACE statements overlap, the resulting trace levels are determined by invoking the TRACE statements in the order in which they appear in the data set.

BUFFERS

Use the BUFFERS statement to specify the parameters used to allocate buffer pools for storing IP datagrams.

Syntax

```
►►──BUFFERS──max_packet_size──┬──────────────────┬──┬──────────────────┬──────────►◄
                              └─add_send_buffers─┘  └─send_queue_limit─┘
```

Parameters

max_packet_size
: The maximum IP packet size. This value should match the *max_packet_size* parameter on the BEGINROUTES or GATEWAY statement in the PROFILE.TCPIP data set and it must be less than the maximum path information unit (PIU) in the VTAM definition for your LU6.2 connection. The maximum PIU is set by the MAXDATA parameter of the PCCU macro that is part of the NCP generation program in the VTAMLST library. The value for

this parameter differs for each device. Check the documentation for your device to determine the appropriate value.

The *max_packet_size* value must be an integer in the range 20 - 32 758. Any datagrams exceeding this length that are received from either the local or remote TCP/IP are discarded.

add_send_buffers
: The number of additional buffers for storing datagrams waiting to be passed to VTAM. These are in addition to the initial allocation of one for every destination (DEST) defined in the SNALINK LU6.2 configuration data set. If an *add_send_buffers* value is not specified, no additional buffers are allocated. The minimum value allowed for *add_send_buffers* is 1; the maximum value allowed is 2 147 483 647.

send_queue_limit
: The maximum number of buffers that can be allocated to any connection for storing datagrams waiting to be passed to VTAM. This parameter allows you to prevent a single connection from restricting other connections' access to the free buffers from the VTAM send buffer pool. Once this maximum limit is reached on the VTAM send queue, further datagrams received from TCP/IP for this connection are discarded, until the number of buffers on the send queue is reduced below the limit. For this parameter to be effective, its value should leave enough free buffers in the VTAM send buffer pool to service the other active connections when full throughput is reached on the main connections. See the description of the *add_send_buffers* parameter to calculate the size of the VTAM send buffer pool. If a *send_queue_limit* value is not specified, no limit is placed on the lengths of individual send queues. The minimum value allowed for *send_queue_limit* is 1; the maximum value allowed is 2 147 483 647.

Examples

```
*-- Buffer specifications:
*
*       Datagram  Add Snd  Snd Queue
*       Max Size  Buffers  Limit
*       -----     -----    -----
BUFFERS 32758     10       11
*
```

DEST

Use the DEST statement to define an IP-address-to-LU-name mapping for a destination node associated with the link specified in the most recent LINK statement.

The IP addresses listed must be consistent with the direct connections defined for the current link in the GATEWAY statement of the *hlq*.PROFILE.TCPIP data set.

Syntax

```
                                           ┌─DATA─┐
►►──DEST──ip_addr──send_lu──receive_lu────┤      ├──────────────────────────────►◄
                                           └─INIT─┘
```

Parameters

ip_addr
: The IP address in dotted decimal format. The value entered must be in the correct format for an IP address of a network node (a fully qualified IP address).

send_lu
: The remote LU name for the *send* connection.

receive_lu
: The remote LU name for the *receive* connection. For independent logical units using parallel sessions, the send and receive LU names are the same. In this case, you can enter an equal sign (=) as the value for the *receive_lu* parameter.

DATA
: Definition of when the connection to the destination node is to be established. If the DATA parameter is specified, the connection is only established after there is IP data to be transferred to and from the destination node. If neither INIT nor DATA is specified, DATA is used as the default setting.

INIT
: Definition of when the connection to the destination node is to be established. If the INIT parameter is specified, the connection is established during initialization of the SNALINK LU6.2 address space.

Examples

```
*        IP Address       Send LU  Recv LU   Type
*        ---------------  -------- --------  ----
DEST     192.9.207.39     LU6LBK11 LU6LBK12  INIT
DEST     192.9.207.40     LU6LBK13 =
DEST     192.9.207.41     LU6LBK14 =
```

LINK

Use the LINK statement to define the link between the main TCP/IP address space and the SNALINK LU6.2 interface.

Syntax

```
►►──LINK──link_name──log_mode──┬─────────────────┬──────────────────────────►◄
                               └─idle_disconnect─┘
```

Parameters

link_name
: The name of the TCP/IP link, as defined in the *hlq*.PROFILE.TCPIP data set, to which the destinations in the subsequent DEST statements are to apply. The maximum length is eight characters.

log_mode
: The name of the VTAM LOGMODE entry to be used when establishing an SNA LU type 6.2 session for this link.

idle_disconnect
: Time, in seconds, after which an idle or inactive session is terminated. If blank or 0, no inactivity checking or timeout is to apply to connections defined for this link. If not blank, the value of this parameter must lie within the inclusive range 0 to 2^{31}–1.

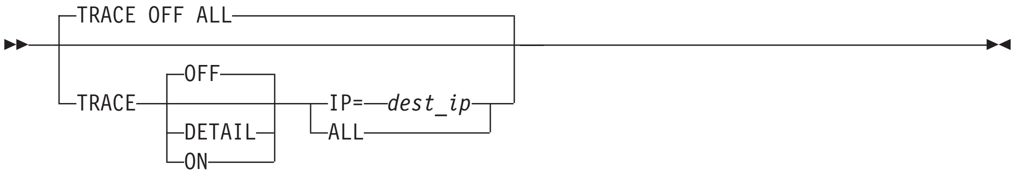
Examples

```
*-- Link Definition:
*                                   Idle
*        TCP/IP Link Name  LogMode   Time-out
*        ----------------  --------  --------
LINK     LNKLU62           LU62MODE  600
```

TRACE

Use the TRACE statement to define the required level of tracing for a specified range of connections. All tracing levels default to OFF unless overridden by any appropriate TRACE statement. Trace output is written to the SYSPRINT data set.

Syntax



Parameters

OFF
Disables tracing for all connections in the specified range. If OFF, DETAIL, or ON is not specified, OFF is the default. For example, specifying `TRACE IP=dest_ip` would disable tracing for only the connection associated with *dest_ip*.

DETAIL
Enables a detailed level of tracing for all connections in the specified range.

ON Enables a basic level of tracing for all connections in the specified range.

IP=*dest_ip*
The destination IP address associated with the connection for which tracing is enabled or disabled. A DEST statement defining a destination IP address must appear before a TRACE statement for the same destination IP address.

ALL
If the ALL parameter is specified, tracing for all destinations (either currently defined or still to be defined) is set to the requested level.

Examples

```
*-- Trace level specifications:
*
*      Trace
*      Level   Connection Range
*      ------  ------------------
TRACE  OFF     ALL
```

VTAM

Use the VTAM statement to specify the VTAM application definition to be used to establish the connections.

The VTAM statement must precede the first LINK statement.

Syntax

```
►►──VTAM──application_id──password──────────────────────────────────────────►◄
```

Parameters

application_id
: The VTAM application identifier as defined by the VTAM APPL statement. This name identifies the local logical unit used by this SNALINK LU6.2 address space. Remote SNALINK LU6.2 interfaces configure this name as their remote logical unit name for a connection to this SNALINK LU6.2 address space.

  The maximum length is eight characters.

password
: The password for the VTAM application specified in *application_id*. This must match the password specified by the PRCT parameter in the VTAM APPL statement.

Examples

The following example shows the connection between the VTAM APPL statement and the VTAM configuration statement.

```
LU62APPL  APPL ACBNAME=LU62APPL,                                       *
               PRTCT=QWERTY,                                           *
```

```
*-- VTAM Application definition:
*
*        ApplID    Password
*        --------  --------
VTAM     LU62APPL  QWERTY
```

Chapter 14. X.25 NPSI

This topic contains the following information

- “X.25 NPSI cataloged procedure (X25PROC)”
- “Sample X.25 NPSI server configuration data set (X25CONF)”
- “Summary of X.25 NPSI server configuration statements” on page 565
- “X.25 NPSI server configuration statements” on page 565

X.25 NPSI cataloged procedure (X25PROC)

This topic shows the X.25 NPSI cataloged procedure (X25PROC).

```
//TCPIPX25 PROC MODULE=XNX25IPI
//*
//* z/OS Communications Server
//* SMP/E Distribution Name: EZAEB020
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5647-A01
//*               (C) Copyright IBM Corp. 1989, 2001
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R2
//*
//X25IPI   EXEC PGM=&MODULE,REGION=256K,TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//X25IPI   DD DSN=TCPIP.SEZAINST(X25CONF),DISP=SHR
//SYSPRINT DD SYSOUT=*
//*
//*        The SYSMDUMP DD statement will cause MVS to provide
//*        an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA
//*        when no GLOBALTCPIPDATA statement is configured.
//*        See the IP Configuration Guide for information on
//*        the TCPIP.DATA search order.
//*        The data set can be any sequential data set or a member of
//*        a partitioned data set (PDS).
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Figure 23. X.25 NPSI cataloged procedure (X25PROC)

Sample X.25 NPSI server configuration data set (X25CONF)

Following is a copy of the sample, X.25 NPSI configuration data set, that is shipped as SEZAINST(X25CONF).

```
*--------------------------------------------------------------------*
*   Sample configuration file for TCPIPX25                           *
*                                                                    *
*   COPYRIGHT = NONE.                                                *
*--------------------------------------------------------------------*
```

```
*-- Trace level and debug flags
*                 01234567
Trace    OFF      00000000
*
*-- VTAM Application definition:
*        ApplID    Password
*        --------  --------
VTAM     TCPIPX25  TCPX25
*
*-----------------------------------------------------------------------*
*   Definitions for a public network connection with two-line hunt
*   group, using NPSI GATE Fast Connect. Network default packet size
*   is 128; packet size 1024 negotiated on call request
*
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU021     3110 23456789        2      128    16
FAST     XU021
Options  PacketSize 1024
AltLink  XU022     3110 34567890        2      128    8
FAST     XU022
Options  PacketSize 1024
Options  AcceptReverse
*
*   Destination address list for this link            Destination
*        IP Address      X.25 DTE Addr       C.U.D.   Facilities
*        --------------- ---------------     -------- ---------------
Dest     192.005.058.001 131106170015300     CC
Dest     192.005.058.004 131106170015320     CC
Dest     192.005.058.005 131106170015350     CC       02AA
*                             * this dest. requires throughput class on
*                                            call request
*
*-----------------------------------------------------------------------*
*   Definitions for a DDN connection
*     Note: DDN and non-DDN links cannot be mixed
*
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
*Link     XU023     DDN                  2       128   16
*Options  GATE
*
*   DDN network 10 uses RFC 1236 address calculation, no explicit
*   X.25 DTE addresses
*
*        IP Address      X.25 DTE Addr
*        --------------- ---------------
*Dest     10
*
*-----------------------------------------------------------------------*
*   Definitions for a private point-to-point link to a router
*
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU024     PRIV 1               2      1024   2
Options  GATE
*
*        IP Address      X.25 DTE Addr       C.U.D.
*        --------------- ---------------     --------
Dest     192.5.57.2      2
*
*-----------------------------------------------------------------------*
*-- Buffer specifications:
*        Datagram  Addn'l    VC Queue
*        Max Size  Buffers   Limit
*        --------  --        --
Buffers  1024      50        4
*
*-- Timers:
*        Idle      Minimum
*        Disconn.  Open
*        --------  -------
Timers   300       60
```

Figure 24. Sample X.25 NPSI server configuration data set (X25CONF)

Summary of X.25 NPSI server configuration statements

Table 29 lists the X.25 NPSI server configuration statements.

Table 29. Summary of X.25 NPSI server configuration statements

Statement	Description	See
ALTLINK	Specifies the members of a link's hunt group for incoming calls	"ALTLINK statement" on page 566
BUFFERS	Specifies the buffer size to use for IP datagrams	"BUFFERS statement" on page 567
DEST	Specifies the destination address list for a link	"DEST statement" on page 568
FAST	Specifies that a link has NPSI GATE fast connect	"FAST statement" on page 569
LINK	Defines the link to an NPSI physical circuit logical unit	"LINK statement" on page 569
OPTIONS	Specifies the call handling options for incoming calls on a link	"OPTIONS statement" on page 570
TIMERS	Defines the time limits for holding or clearing connections on all links	"TIMERS statement" on page 572
TRACE	Specifies the trace and debug levels for the X.25 NPSI server	"TRACE statement" on page 572
VTAM	Identifies the VTAM APPL definition for the X.25 NPSI server	"VTAM statement" on page 574

X.25 NPSI server configuration statements

Following are the syntax and description of the valid statements used in the data set pointed to by the //X25IPI DD statement in your X.25 NPSI cataloged procedure.

Statement syntax

Rules: Follow these statement syntax rules:

- Each statement is on a separate line in the configuration data set.
- Each statement starts with the keyword followed by the parameter fields, separated by one or more blanks.
- The statements are not case sensitive. You can enter them in both upper- and lowercase.
- Comment lines are marked with an asterisk (*) in column 1.

ALTLINK statement

Use the ALTLINK statement to specify members of a hunt group for incoming calls. The collection of lines in this group is assigned a single X.25 address. Incoming X.25 calls are accepted from any of the lines in the group and outgoing calls are rotated across the lines. If one of the lines is not operational, outgoing calls are rotated on to the next available line in the group.

Syntax

```
►►──ALTLINK──mchlu_name──┬─DDN─────────────┬──────────────────────────────────────►
                         └─dnic dte_addr───┘

►─window_size──packet_size──logical_channels───────────────────────────────────►◄
```

Parameters

mchlu_name
   The name of the physical circuit logical unit (NPSI MCH LU).

DDN
   Use DDN for the Defense Data Network.

dnic
   The X.121 Data Network Identifier Code (DNIC) for the public data network. *dnic* can be coded as PRIVATE or PRIV to denote a private X.25 network.

dte_addr
   The X.25 DTE address for the link. The address must be from 1 - 15 decimal digits. This parameter is not coded for DDN links. Specify *dte_addr* as NONE to omit the calling address from the call request packet.

window_size
   The window size to negotiate on switched virtual circuits, in the range of 1 - 7 for a modulo-8 network, or 1 - 127 for a modulo-128 network.

packet_size
   Choose one of the following X.25 packet sizes as the default: 32, 64, 128, 256, 512, 1 024, 2 048, or 4 096 bytes.

logical_channels
   The number of logical channels (switched virtual circuits) subscribed, in the range of 1 to 1 023.

Examples

The following example shows the LINK, ALTLINK, FAST, and OPTIONS statements for a public network connection with a two-line hunt group:

```
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU021     3110 23456789        2      128    16
FAST     XU021
Options  PacketSize 1024
AltLink  XU022     3110 34567890        2      128    8
FAST     XU022
Options  PacketSize 1024
Options  AcceptReverse
```

Usage notes

- The ALTLINK statement must follow a LINK statement.
- A DEST statement follows the last ALTLINK statement for a hunt group.
- If special options are required or fast connect is used, OPTIONS or FAST statements must immediately follow the ALTLINK to which they apply.

Related topics

- "DEST statement" on page 568
- "FAST statement" on page 569
- "LINK statement" on page 569

BUFFERS statement

Use the BUFFERS statement to specify the buffer size for IP datagrams.

Syntax

```
►►──BUFFERS──max_packet_size──add_buffers──vc_queue_limit──────────────────────►◄
```

Parameters

max_packet_size
: The maximum IP packet size. This value must match the *max_packet_size* parameter on the GATEWAY statement in the *hlq*.PROFILE.TCPIP data set, and must be in the range of 576 - 2 048.

add_buffers
: The number of additional buffers to allocate, in addition to the minimum of 2 for each logical channel.

vc_queue_limit
: The limit on the number of buffers queued outbound on any single virtual circuit.

Examples

The following statement specifies a maximum IP packet size of 1024, an allocation of 50 additional buffers, and a limit of 4 queued outbound buffers on any SVC:

```
*
Buffers  1024     50      4
*
```

Usage notes

- This maximum IP packet size must be at least as large as the largest *max_packet_size* parameter on the GATEWAY entries for X.25 NPSI LINKS in the *hlq*.PROFILE.TCPIP data set.
- Additional buffers are required for coping with traffic peaks and holding outbound IP datagrams while new X.25 connections are being established. Use a larger value when many X.25 destinations are called in a short period of time.

Related topics

- "GATEWAY statement" on page 109
- "OPTIONS statement" on page 570

- z/OS Communications Server: IP Diagnosis Guide

DEST statement

Use one or more DEST statements to specify the destination address list for the link.

Syntax

```
►►──DEST──ip_addr──┬──────────────┬──┬─────┬──┬──────────────────┬──────────────►◄
                   └─X25_dte_addr─┘  └─cud─┘  └─dest_facilities─┘
```

Parameters

ip_addr
: The IP address in dotted decimal format. At least 1 byte must be supplied; omitted trailing bytes are not checked when determining a match.

X25_dte_addr
: The corresponding X.25 DTE destination address for the link (1 to 15 decimal digits). Do not code this parameter for Defense Data Network (DDN) destinations.

cud
: The call user data (CUD) protocol identifier used. A hexadecimal number with a default value of X'CC'. Do not code this parameter for DDN destinations.

dest_facilities
: The X.25 facilities field to be used on outgoing calls for this destination. This value overrides the FACILITIES value in the OPTIONS statement for this destination. Specify this value as an even number of hexadecimal digits. The field is inserted in outgoing call packets following facilities generated from window or packet size negotiation or reverse charging. The facilities length byte is calculated automatically and should not be coded here.

Examples

The following example shows the LINK and DEST statement for a DDN connection.

```
*        NPSI MCH       DTE             Window Packet Logical
*        LU Name   DNIC Address         Size   Size   Channels
*        --------  ---- --------------- -      ----   ---
Link     XU023     DDN                  2       128   16
*
*        IP address      X.25 DTE addr
*        --------------- ---------------
Dest     10
*
```

Usage notes

- The DEST statements must follow the LINK statement.
- Data Defense Network destinations do not use the X.25 DTE address and the CUD protocol identifier.

FAST statement

Use the FAST statement to provide NPSI fast connect for links with heavy activity. Fast connect is only used for SVCs connected to non-SNA data terminal equipment (DTE). See *X.25 NPSI Host Programming* for more information.

```
                   ┌─HEX─┐   ┌─001────┐
►►──FAST──prefix───┼─────┼───┼────────┼──────────────────────────────────────────►◄
                   └─DEC─┘   └─suffix─┘
```

Parameters

prefix
: The fast connect VC LU name prefix. This is the MCH LU name unless the PRFLU option is coded on the NPSI X25.VC statement.

HEX or DEC
: The fast connect VC LU numbering scheme (if the HEXNAME parameter is coded in the NPSI X25.VC statement). The default is HEX.

suffix
: The fast connect VC LU numbering base (if the SUFFIX parameter is coded in the NPSI X25.VC statement). The default is 001.

Examples

The following example shows the placement of a FAST statement that specifies a *prefix* of XU021 and takes the default values of HEX and 001.

```
*       NPSI MCH       DTE             Window Packet Logical
*       LU Name   DNIC Address         Size   Size   Channels
*       --------  ---- --------------- -      ----   ---
LINK    XU021     3110 23456789        2      128    16
FAST    XU021
OPTIONS  GATE
*
```

Usage notes

- The OPTIONS GATE statement is required.
- The FAST statement must follow the LINK or ALTLINK statement to which it applies.
- The *prefix* value must match the value in your NPSI configuration.

LINK statement

Use the LINK statement to define the NPSI MCH LU names. One SNA control session is established for each MCH LU defined by a LINK statement.

Syntax

```
►►──LINK──mchlu_name───┬─DDN────────────┬─────────────────────────────────────►
                       └─dnic dte_addr──┘

►─window_size──packet_size──logical_channels──────────────────────────────────►◄
```

Parameters

mchlu_name
The name of the physical circuit logical unit (NPSI MCH LU).

DDN
Use DDN for the Defense Data Network.

dnic
The X.121 Data Network Identifier Code (DNIC) for the public data network. The *dnic* parameter can be coded as PRIVATE or PRIV to denote a private X.25 network.

dte_addr
The X.25 DTE address for the link. The address must be from 1 - 15 decimal digits. This parameter is not coded for DDN links. Specify *dte_addr* as NONE to omit the calling address from the call request packet.

window_size
The window size to negotiate on switched virtual circuits, in the range of 1 - 7 for a modulo-8 network, or 1 - 127 for a modulo-128 network.

packet_size
Choose one of the following X.25 packet sizes as the default: 32, 64, 128, 256, 512, 1 024, 2 048, or 4 096 bytes.

logical_channels
The number of logical channels (switched virtual circuits) subscribed, in the range of 1 - 1 023.

Examples

In the following example, AU20 is the name of a non-DDN network, and AU16 is the name of a DDN network.

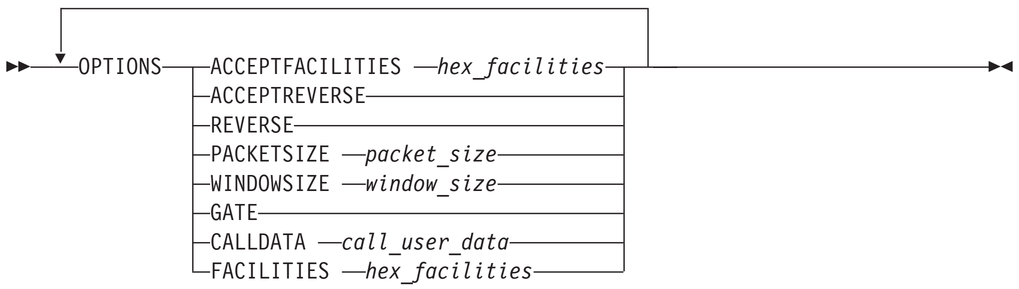
```
*        NPSI MCH        DTE             Window Packet Logical
*        LU Name   DNIC  Address         Size   Size   Channels
*        --------  ----  ---------------  -     ----    ---
LINK     AU20      3020  90201234548      2     1024     5
LINK     AU16      DDN                    2     1024     5
*
```

OPTIONS statement

Use the OPTIONS statement to specify the call handling options for each link. Values specified on the OPTIONS statement apply to all outgoing calls on the LINK MCH, but can be overridden for individual destination addresses by the DEST statement. More than one OPTIONS statement can be coded after each LINK statement.

Restriction: Several parameters can be placed in a single OPTIONS statement, but cannot continue on the next line. If all the parameters do not fit on one line, use additional OPTIONS statements.

Syntax



Parameters

ACCEPTFACILITIES *hex_facilities*
The X.25 facilities field to be used when accepting incoming calls. Specify this value as an even number of hexadecimal digits. The facilities length byte is calculated automatically and should not be coded here.

ACCEPTREVERSE
Causes incoming calls with the reverse charging facility to be accepted. The default action is to clear reverse charge calls.

REVERSE
Includes the reverse charging facility in all outgoing call request packets.

PACKETSIZE *packet_size*
The packet size to negotiate on switched virtual circuits, one of the values 32, 64, 128, 256, 512, 1 024, 2 048, or 4 096 bytes.

WINDOWSIZE *window_size*
The window size to negotiate on switched virtual circuits, in the range of 1 - 7 for a modulo-8 network, or 1 - 127 for a modulo-128 network.

GATE
Specified if the NPSI MCH is defined with GATE=GENERAL to permit sharing of an X.25 physical link with other services.

CALLDATA *call_user_data*
The call user data field to be used on outgoing X.25 calls, specified as an even number of hexadecimal digits. The standard value for IP traffic must begin with the protocol identifier CC.

FACILITIES *hex_facilities*
The X.25 facilities field to be used on outgoing calls for this destination. Specify this value as an even number of hexadecimal digits. The field is inserted in outgoing call packets following facilities generated from window or packet size negotiation or reverse charging. The facilities length byte is calculated automatically and should not be coded here.

Examples

The following example shows the proper placement of the OPTIONS statements when using both LINK and ALTLINK statements.

```
Link     XU021     3110 23456789       2      128    16
FAST     XU021
Options  PacketSize 1024
AltLink  XU022     3110 34567890       2      128    8
FAST     XU022
Options  PacketSize 1024
Options  AcceptReverse
*
```

Usage notes

- The OPTIONS statements must follow the LINK or ALTLINK statements to which they apply.
- Negotiation takes place on outgoing calls if the window size or packet size on the OPTIONS statement is different from the network defaults coded in the LINK statement.
- The *max_packet_size*, also called the maximum transmission unit (MTU), coded in the BUFFERS statement must be large enough to hold the largest IP datagram to be transmitted or received over the link. If the MTU is greater than the X.25 packet size, an IP datagram is sent as an X.25 packet sequence. The buffer size must be sufficient to hold the combined data of the sequence. The MTU for DDN networks is 1007. See RFC 877 for more information. information about how to obtain RFCs is included in z/OS Communications Server: IP Configuration Guide.

Related topics

- "BUFFERS statement" on page 567
- "GATEWAY statement" on page 109

TIMERS statement

Use the TIMERS statement to specify the limits for various timers.

Syntax

```
►►──TIMERS──idle_disconnect──min_open────────────────────────────────────►◄
```

Parameters

idle_disconnect
Time, in seconds, after which an idle or inactive connection is cleared.

min_open
The minimum time, in seconds, a connection is held before it can be preempted by a new destination.

Examples

The following statement clears inactive connections after 5 minutes and holds a connection open for 1 minute.

```
*        Idle      Minimum
*        Disconn.  Open
*        --------  -------
Timers   300       60
*
```

TRACE statement

Use the TRACE statement to specify the trace and debug levels for the X.25 NPSI server. The trace and debug functions are independent of one another. You can turn tracing off and still request debug options.

Syntax

```
►►──TRACE──┬─OFF─────┬─debug_flags──────────────────────────────────────────►◄
           ├─DATA────┤
           ├─CONTROL─┤
           ├─EBCDIC──┤
           ├─ASCII───┤
           └─IA5─────┘
```

Parameters

OFF
Turns tracing off. If you specify OFF, data on the connection is not traced.

DATA
Traces the data packets on the connection and displays the full contents of the IP datagrams. This is equivalent to the ASCII option.

CONTROL
Traces the data packets on the connection and displays only the X.25 control packet.

EBCDIC
Traces the data packets on the connection and displays the data in EBCDIC.

ASCII
Traces the data packets on the connection and displays the data in ASCII.

IA5
Traces the data packets on the connection and displays the data in IA5.

debug_flags
A string of eight positional flags that control the display of debugging information. Each flag has the value of 1 or 0, where 1 turns the flag on and 0 turns the flag off. The flags are:

Position	**Description**
0	Display configuration records
1	Display commands
2	Trace DLC events
3	Trace VTAM events
4	Display control block addresses
5	Main loop dispatching
6	Reserved for internal use
7	Send information and warning messages to the operator console

Examples

The following statement turns tracing off and sets two debug flags:

Value 1 in position 0
Displays configuration records

Value 1 in position 7
Sends information and warning messages to the operator console

```
*                   01234567
Trace    OFF        10000001
*
```

Related topics

See z/OS Communications Server: IP Diagnosis Guide for more information.

VTAM statement

Use the VTAM statement to access the VTAM definition for the application. The VTAM statement must precede the LINK statement.

Syntax

```
►►──VTAM──application_id──password─────────────────────────────────────────────►◄
```

Parameters

application_id
The application identifier in the VTAM APPL definition. This is either the name specified in the first 8 columns or the ACBNAME if one is defined.

password
The password for the VTAM application specified in the VTAM APPL definition.

Examples

This VTAM statement is correct for either of the VTAM APPL definitions that follow it.

```
*--
VTAM     TCPIPX25  TCPX25
*--
```

VTAM APPL definitions:

```
X25APPL2 APPL  ACBNAME=TCPIPX25,
               PRTCT=TCPX25,
               AUTH=(ACQ),
               PARSESS=YES,
               EAS=20

TCPIPX25 APPL  PRTCT=TCPX25,
               AUTH=(ACQ),
               PARSESS=YES,
               EAS=20
```

Chapter 15. NCPROUTE server

This topic includes the following information:
- "NCPROUTE cataloged procedure (NCPROUT)" on page 582
- "Specifying the NCPROUTE parameters" on page 583
- "NCPROUTE profile data set" on page 584
- "NCPROUTE gateways statements and syntax rules"

Restriction: OSPF and IPv6 is not supported for NCPROUTE.

Related topics

See the following topics for more information:
- "BEGINROUTES statement" on page 28
- "BSDROUTINGPARMS statement" on page 36
- "DEVICE and LINK — SNA LU0 links statement" on page 95
- "DEVICE and LINK — 3745/46 channel DLC devices statement" on page 105
- "GATEWAY statement" on page 109
- "HOME statement" on page 136
- "IPCONFIG statement" on page 190
- Chapter 4, "Protocol number and port assignments," on page 347
- Chapter 11, "OMPROUTE," on page 483
- Chapter 12, "SNALINK," on page 553
- z/OS Communications Server: IP Configuration Guide
- z/OS Communications Server: IP Diagnosis Guide
- z/OS Communications Server: IP System Administrator's Commands

NCPROUTE gateways statements and syntax rules

This topic includes the NCPROUTE gateways statements.

Observe the following rules:
- You can specify multiple GATEWAY statements.
- Keywords can be specified in mixed case.
- Blanks and comments are supported in the gateways data set. Comments are identified by a semicolon (;) as the first non-whitespace character in a file record.
- GATEWAY statements can start in any column, but they cannot wrap from one file record to the next.
- There should be no sequence numbers in the data set.

Table 30 on page 576 shows how the selected parameters affect the advertising algorithm for routes in RIP responses to adjacent routers. The parameters can be used as router-wide RIP output filters. To configure interface-wide RIP input and output filters, see the OPTIONS statement in the NCPROUTE Gateways configuration data set.

Table 30. NCPRoute parameters and options

Parameter or Option	Host routes (direct or indirect)	Direct (local) network routes	Indirect network routes	Default routes	Unreachable routes
default.router		Yes	Yes	Yes	Yes
Supply default.route				Yes	Yes
-h or supply hosts	Yes	Yes	Yes		Yes
-sl or supply locals		Yes			Yes
-s or supply on		Yes	Yes		Yes
-sq or supply off					
None		Yes	Yes		Yes

GATEWAY statement

This topic describes how to use the GATEWAY statement.

Syntax

```
►►──┬─net────┬──name1──gateway──name2──metric──value──┤ gateway options ├──────►◄
    ├─host───┤
    └─active─┘
```

gateway options:

```
├──┬─passive──┬──mask──subnetmask──────────────────────────────────────────┤
   ├─external─┤
   └─active───┘
```

Parameters

net
Indicates that the route goes to a network.

host
Indicates that the route goes to a specific host.

active
Indicates that the route to the gateway is treated as a network interface.

name1
Either a symbolic name or the IP address of the destination network or host.

Restriction: If this is for an active gateway, *name1* must be specified as `active`.

gateway
The parameters that follow this keyword identify the gateway or router for this destination.

name2
Either a symbolic name or the IP address of the gateway or router for this destination.

metric
The value that follows this keyword is the hop count to the destination.

value
Indicates the hop count to this destination. This number is an integer from 1 to 15, where 15 indicates that the network cannot be reached.

passive
A passive gateway does not exchange routing information. Information about the passive gateway is maintained in the local routing tables indefinitely and is only local to this NCPROUTE server. Passive gateway entries for indirect routes are not included in any routing information that is transmitted. Directly connected passive routes are included.

external
Indicates that entries for this destination should never be added to the routing table. The NCPROUTE server discards any routes for this destination that it receives from other routers. Only the destination field is significant; the gateway and metric fields are ignored.

active
Indicates that the router is treated as a network interface. An active gateway is a router that is running RIP, but can only be reached through a network that does not allow link-level broadcasting or multicasting and is not point-to-point.

mask
A constant. The value that follows this keyword is the subnet mask for the route.

subnetmask
A bit mask (expressed in dotted-decimal form) defining the subnetwork mask for a network route. If the *subnetmask* is not specified, NCPROUTE defaults the subnetwork mask to an interface subnetwork mask that matches the route's network. If there is no interface match, then the network class mask for the route is used.

Requirement: The bits must be contiguous in the network portion of the *subnetmask*.

OPTIONS statement

This topic describes how to use the OPTIONS statement.

Syntax

```
   ┌─────────────────────────────────────────────────────────────┐
   ▼                         ┌─no──┐                              │
►►──OPTIONS──┬─default.router─┴─yes─┴──────────────────────────┬─┴──────────►◄
             │          ┌─on──────────────┐                     │
             ├─supply───┼─off─────────────┼─────────────────────┤
             │          ├─default.route───┤                     │
             │          ├─hosts───────────┤                     │
             │          └─locals──────────┘                     │
             │              ┌─0─┐                               │
             ├─trace.level──┼─1─┼───────────────────────────────┤
             │              ├─2─┤                               │
             │              ├─3─┤                               │
             │              └─4─┘                               │
             ├─gateway ──ip_addr ──┬─block ──────┬──────────────┤
             │                     ├─noreceive───┤              │
             │                     └─none────────┘              │
             └─interface ──name ──ip_addr ──┤ interface options ├─┘
```

interface options:

```
├──┬─block ──destination ──fmask ──mask ───────────┬──────────────────────────┤
   ├─forward ──destination ──fmask ──mask ─────────┤
   ├─forward.cond ──destination ──fmask ──mask ────┤
   ├─metric ───────────────────────────────────────┤
   ├─noforward ──destination ──fmask ──mask ───────┤
   ├─receive ──destination ──fmask ──mask ─────────┤
   ├─receive.cond ──destination ──fmask ──mask ────┤
   ├─noreceive ──destination ──fmask ──mask ───────┤
   ├─none──────────────────────────────────────────┤
   ├─passive ──────────────────────────────────────┤
   ├─ripon ────────────────────────────────────────┤
   ├─ripoff ───────────────────────────────────────┤
   │          ┌─off─┐                              │
   ├─supply ──┴─on ─┴──────────────────────────────┤
   ├─key──authentication_key ──────────────────────┤
   ├─nokey─────────────────────────────────────────┤
   ├─supply.control──supply_control ───────────────┤
   └─receive.control──rec._control ────────────────┘
```

Parameters

`default.router`
Enables the default router. When this option is specified, NCPROUTE adds a default route to its routing information and propagates it over all local interfaces. If the adjacent routers add the default route to their routing tables, NCPROUTE receives all unknown packets from them and funnel them to a destination router, provided that a default route is defined. If this option is used, define a default route to a destination router on an IPROUTE statement in NCP generation definition or in the NCPROUTE gateways data set. See z/OS Communications Server: IP Configuration Guide.

yes This is the default router.

no This is not the default router.

`interface`
A constant. The parameters *name* and *ipaddr* follow this keyword.

name Specifies the name of the interface according to NCP clients-NCP generation. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all interface names.

ipaddr Specifies the IP address of the interface associated with the interface name. A specification of an asterisk (*) can only be used with the NONE parameter option to indicate all IP addresses of the interfaces.

noreceive (or block)
If an interface option, specifies that the *destination* route in the received RIP packets for this interface are to be ignored. If a gateway option, specifies that routing table broadcasts from this gateway are to be ignored. This option is provided as an RIP input filter.

destination
Specifies that the destination route is in network, subnetwork, or host format. A specification of an asterisk (*) indicates that all destination routes are to be used with the *noforward* and *noreceive* options. This serves as a blackhole filter option which can be used to filter out all routes RIP packets over an interface and allow routes with specified *forward* and *receive* filters to be used.

fmask A constant. The value that follows this keyword is the filter mask for the route.

mask Optional bit mask (expressed in dotted-decimal form) defining the routing filter mask associated with the destination route. This mask is to be used as an optional parameter to the forward and receive parameters to filter in and filter out multiple routes matching the mask of the destination route. This option can be used to define a single RIP input or output filter representing multiple routes as opposed to defining individual RIP input or output filters for each route.

forward
Specifies that the *destination* route in the RIP responses is to be forwarded to this interface only. This option is provided as an RIP output filter and can be used for inbound and outbound traffic splitting.

forward.cond
Specifies that the *destination* route is to be forwarded to this interface only when the interface is active. In case of an interface outage, NCPROUTE includes the *destination* route in the RIP responses to other active interfaces. After recovery of an interface outage, NCPROUTE resumes to sending the destination route over this interface only. This option is provided as an RIP output filter and can be used for inbound and outbound traffic splitting.

metric The metric associated with the cost of use for the link. When sending routing information over this link, NCPROUTE uses the *new_metric* value in the routing metrics for the routes that are advertised over this link. If this option is not used, the metric value that is used is the value specified in the IPLOCAL statement of NCP generation definition. This option allows you to override the genned metric. If a metric of 1 is specified, a metric value of 1 is used; this is the default cost for a directly connected network. If a metric of 2 is specified, a metric value of 2 is used. As the metric gets higher, the routes sent over this link become less preferred. The range is from 1 to 15. A metric of 1 is usually coded so that the routes sent over the interface is the most preferred.

noforward
Specifies that the destination route in the RIP responses is not to be forwarded. This option is provided as an RIP output filter.

none If an interface option, specifies that any RIP filter options for this interface are to be turned off or reset. If asterisks (*) are specified for interface name and ipaddr, all options are cleared from all interfaces. If a gateway option, specifies that any RIP filter options for the gateway are to be turned off or reset. If an asterisk (*) is specified for the IP addresses, all gateway entries with gateway options are cleared.

receive
Specifies that the destination route in the RIP responses is to be received over this interface only. This option is provided as an RIP input filter.

receive.cond
Specifies that the destination route is to be received over this interface only when it is active. In case of an interface outage, NCPROUTE allows the destination route in the RIP responses to be received over other active interfaces. This option is provided as an RIP input filter and can be used for inbound and outbound traffic splitting.

ripoff (or passive)
Specifies that RIP is disabled for this interface. NCPROUTE does not supply nor receive RIP updates.

ripon Specifies that RIP is enabled for this interface. RIP responses are allowed to be sent or received over this interface.

`supply`
Defines the supply routing setting. The default is on. This option is provided as an RIP input and output filter.

on Supply routing information for this NCP client or interface.

off Suppresses supply of routing information for this NCP client or interface. NCPROUTE continues to receive routing updates.

default.route
Supply the default route only for this NCP client. When this option is specified, *yes* is internally set for the *default.router* option. This option is provided as an RIP output filter.

hosts Supply routing information with host routes added.

locals Supply only local (directly connected) routes.

`trace.level`
Specifies the trace level to be used for this NCP client. The default is 0.

0 Do not allow tracing.

1 Activates tracing of actions by the NCPROUTE server.

2 Activates tracing of actions and packets sent or received.

3 Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced and are displayed whenever an interface goes inactive.

4 Activates tracing of actions, packets sent or received, packet history, and packet contents. The packet contents display the RIP network routing information.

`key`
Specifies a plain text password authentication key containing up to 16 characters to be used for this interface and that is used to override the server-wide setting defined in the NCPROUTE profile. It can contain mixed case and blank characters. Single quotation marks (') can be included as delimiters to include leading and trailing blanks. A null or blank key indicates that the server-wide key is used as the default. For examples on authentication passwords, see the RIP2_AUTHENTICATION_KEY statement in "NCPROUTE profile data set" on page 584.

`nokey`
Specifies that authentication is disabled for this interface even though the server-wide specification from the NCPROUTE profile is defined.

`supply.control`
Specifies that the RIP *supply_control* is to be used for this client or interface and is used to override the NCPROUTE profile setting.

`supply_control`
Specifies one of the RIP supply control options. The default is set to the NCPROUTE profile setting. The following values are valid options:

- RIP1
- RIP2B
- RIP2M
- RIP2
- NONE

`receive.control`
Specifies that the RIP *receive_control* is to be used for this client or interface and is used to override the NCPROUTE profile setting.

`rec_control`
Specifies one of the RIP receive control options. The default is set to the NCPROUTE profile setting. Valid options are RIP1, RIP2, ANY, and NONE.

`gateway`
A constant. The value that follows this keyword identifies the gateway or router.

`ipaddr`
If an interface option, specifies the IP address of the interface associated with the interface name. If a gateway option, specifies the gateway address of the adjacent router. A specification of an asterisk (*) applies to all gateway addresses.

`none`
If an interface option, specifies that any RIP filter options for this interface are to be turned off or reset. If a gateway option, specifies that any RIP filter options for this gateway are to be turned off or reset. A specification of an asterisk (*) indicates all interface IP addresses or all gateway addresses.

`noreceive (or block)`
If an interface option, specifies that the *destination* route in the RIP responses propagates is not to be received over this interface only. If a gateway option, specifies that no RIP packets are to be received from the specified gateway address of the adjacent router. This option provides an RIP input filter.

Result: All traces go to a standard output referred to in the //SYSPRINT DD statement in the NCPROUTE cataloged procedure.

The options can be specified in any order. For example:

```
   options default.router yes supply on trace.level 2
   options interface ETH1 10.1.1.1 passive
   options interface ETH1 10.1.1.1 supply off
   options interface TR1 9.67.112.25 metric 2
   options interface TR1 9.67.112.25 forward 11.0.0.0
   options interface TR1 9.67.112.25 forward.cond 12.0.0.0
   options interface TR1 9.67.112.25 block 9.1.0.0
   options interface TR1 9.67.112.25 supply.control rip1
   options interface ETH1 10.1.1.1 receive.control rip2
   options interface ETH2 9.1.1.1 forward 9.2.0.0 fmask 255.255.0.0
   options interface ETH1 10.1.1.1 none
   options interface * * none
   options gateway 9.67.112.77 noreceive
   options gateway 9.67.112.77 none
   options gateway * none
```

NCPROUTE cataloged procedure (NCPROUT)

This topic shows the NCPROUTE cataloged procedure (NCPROUT).

```
//NCPROUT PROC MODULE=NCPROUTE,PARMS='/'
//*
//*  z/OS Communications Server
//*  SMP/E Distribution Name: EZBNRJCL
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5647-A01
//*               (C) Copyright IBM Corp. 1994, 2001
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R2
//*
//NCPROUT  EXEC PGM=&MODULE,
//            PARM='&PARMS',
//            REGION=4096K,TIME=1440
//*
//*  STEPLIB contains libraries to be accessed by NCPROUT.  Required
//*  libraries are the TCPIP executable module library and the NCP
//*  load library which contains a client's NCP load module and its
//*  Routing Information Table (RIT).
//*
//*  The C runtime libraries should be in the system's link list or add
//*  them to the STEPLIB definition here.  If you add them to STEPLIB,
//*  they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*        DD DSN=ncp.v7r1.ncpload,DISP=SHR
//*
//*  SYSPRINT contains output from NCPROUTE plus any enabled tracing.
//*  It can be a data set or SYSOUT.
//*
//SYSPRINT DD SYSOUT=*
//*
//*  SYSERR contains abnormal run-time error messages from NCPROUTE.
//*  It can be a data set or SYSOUT.
//*
//SYSERR   DD SYSOUT=*
//*
//*  The SYSMDUMP DD statement will cause MVS to provide
//*  an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*  NCPRPROF contains profile configuration information such as SNMP
//*  agent specifications and the name of a GATEWAYS partitioned data
//*  set (PDS).  A data set member can be optionally created for each
//*  NCP client and it can contain gateway definitions and NCPROUTE
//*  server options such as tracing and broadcasting of route tables.
//*
//*  The data set can be any sequential data set or a member of a
//*  partitioned data set (PDS).  For a sequential data set, specify
//*  FREE=CLOSE parameter for dynamic allocation support.
//*
//NCPRPROF DD DSN=TCPIP.SEZAINST(EZBNRPRF),DISP=SHR
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA
//*        when no GLOBALTCPIPDATA statement is configured.
//*        See the IP Configuration Guide for information on
//*        the TCPIP.DATA search order.
//*        The data set can be any sequential data set or a member of
//*        a partitioned data set (PDS).
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
//*
//*  SERVICES points to an optional etc.services data set which can be
//*  used to override well-known ports in the ETC.SERVICES.
//*
//*SERVICES DD DSN=TCPIP.SEZAINST(EZAEB02J),DISP=SHR
//*
//*  MSNCPROU contains NCPROUTE's optional message repository for NLS
//*  support.
//*
//*MSNCPROU DD DSN=TCPIP.SEZAINST(EZBNRMSG),DISP=SHR,FREE=CLOSE
```

Figure 25. NCPROUTE catalogued procedure

Specifying the NCPROUTE parameters

The system parameters required by NCPROUTE are passed by the PARM parameter on the EXEC statement of the NCPROUTE cataloged procedure. Add your parameters to `PARMS='/'` in the PROC statement of the NCPROUTE cataloged procedure, making certain that:

- A slash (/) precedes the first parameter.
- Each parameter is separated by a blank.
- Mixed case is allowed for the parameters.
- Blanks and comments are supported in the gateways data set. Comments are identified by a semicolon (;).

For example: `//NCPROUT PROC MODULE=NCPROUT,PARMS='/-s -t -t'`

Guideline: These parameters are also valid when starting the NCPROUTE server with the START command or when modifying NCPROUTE with the MODIFY command. For more information about parameters used with the MODIFY command, see z/OS Communications Server: IP System Administrator's Commands.

Parameters

-dp
: Trace packets coming in and out of NCPROUTE for all NCP clients. The packets are displayed in data format.

-h Include host routes in the RIP responses. Adjacent routers to an NCP client must be able to receive host routes. Otherwise, NETWORK UNREACHABLE problems occur.

-s Supply routing information for all NCP clients and override the supply settings in the NCP clients' gateways data sets.

-sl
: Supply local (directly connected) routes only for NCP clients. This option is provided as an RIP output filter.

-sq
: Suppress supplying routing information to all NCP clients and override the supply settings in the NCP clients' gateways data set.

-t
: Activate global tracing of actions for all NCP clients.

-tq
: Deactivate tracing at all levels. This parameter suppresses tracing for all NCP clients and overrides the trace settings in the NCP clients gateway data set.

-t -t
: Activate global tracing of packets for all NCP clients.

Guideline: There are no third or fourth level global tracing options like those in the NCPROUTE gateways data set members. However, additional levels can be specified using the MODIFY command for a specific NCP client. In any case, the system uses the highest setting.

For more information, see z/OS Communications Server: IP Configuration Guide.

All traces go to a standard output referred to by the //SYSPRINT DD statement in the NCPROUTE cataloged procedure. All abnormal runtime error messages go to the data set specified by the //SYSERR DD statement in the NCPROUTE cataloged procedure.

NCPROUTE profile data set

To build the NCPROUTE profile, create a data set and specify its name in the //NCPRPROF DD statement in the NCPROUTE cataloged procedure. A sample is in SEZAINST(EZBNRPRF). Include configuration statements in this data set to define SNMP functions and to identify the NCPROUTE gateways data set. For more information about configuring SNMP, see z/OS Communications Server: IP Configuration Guide.

The following statements can be included in the NCPROUTE profile:

RIP_SUPPLY_CONTROL *supply_control*
: Specifies one of the following options on a server-wide basis:
- RIP1—Unicast/Broadcast RIP Version 1 packets (Default)
- RIP2B—Unicast/Broadcast RIP Version 2 packets (Not preferred)
- RIP2M—Unicast/Multicast/Broadcast RIP packets (Migration)
- RIP2—Unicast/Multicast RIP Version 2 packets
- NONE—Disables sending RIP packets

Guidelines:

- If RIP2 is specified, the RIP Version 2 packets are multicast over multicast-capable interfaces only. No RIP packets are sent over multicast-incapable interfaces.
- For RIP2M, the RIP Version 2 packets are multicast over multicast-capable interfaces and RIP Version 1 packets over multicast-incapable interfaces.
- For RIP2B, the RIP Version 2 packets are unicast or broadcast; do not use this option because host route misinterpretations by adjacent routers running RIP Version 1 can occur. For this reason, RIP2B might become obsolete in a future release. For point-to-point interfaces that are nonbroadcast and multicast-incapable, the RIP Version 2 packets are unicast.

`RIP RECEIVE CONTROL` *`receive_control`*
Specifies one of the following options on a server-wide basis:

- RIP1—Receive RIP Version 1 packets only
- RIP2—Receive RIP Version 2 packets only
- ANY—Receive any RIP Version 1 and 2 packets (Default)
- NONE—Disables receiving RIP packets

Restriction: If the client NCP does not support variable subnetting, the default of ANY is changed to RIP1.

`RIP2_AUTHENTICATION_KEY` *`authentication_key`*
Specifies a plain text password *authentication_key* containing up to 16 characters. The key is used on a router-wide basis and can contain mixed case and blank characters. Single quotation marks (') can be included as delimiters to include leading and trailing blanks. The key is used to authenticate RIP Version 2 packets and be included in the RIP updates for authentication by adjacent routers running RIP Version 2. For maximum security, set RIP_SUPPLY_CONTROL and RIP RECEIVE CONTROL to RIP2. This discards RIP1 and unauthenticated RIP2 packets. A blank key indicates that authentication is disabled. Following are examples of authentication passwords:

```
my password        (no leading or trailing blanks)
' my password '    (leading and trailing blanks)
''abc''            (single quotes part of password)
'     '            (5-character blanks)
```

`SNMP_AGENT` *`host_name`*
Specifies the host name or IP address of the host running an SNMP daemon.

Restriction: Only one NCPROUTE server can use a particular SNMP agent at a time.

`SNMP_COMMUNITY` *`community_name`*
Specifies a community name that SNMP applications must use to access data that the agent manages. Protect this information accordingly.

`GATEWAY_PDS` *`dsname`*
Specifies the optional partitioned data set that contains GATEWAY information for each client NCP. Quotation marks are not needed when specifying *dsname*. One member for each NCP client of this data set must be configured to match the NCP NEWNAME parameter with the **P** suffix, which is the same as the NCP's RIT member name. See the information about configuring NCPROUTE gateways in z/OS Communications Server: IP Configuration Guide for additional information about defining the statements necessary for the members of this data set.

Chapter 16. TN3270E Telnet server

This topic describes the TN3270E Telnet server (Telnet) parameter and mapping statements.

Telnet profile statements overview

These statements define the characteristics of connections, which host VTAM applications can be accessed, what LU name represents the client, and other functions. For a detailed discussion of Telnet functions, see z/OS Communications Server: IP Configuration Guide.

TELNETGLOBALS statements

The TELNETGLOBALS block is an *optional* statement block that contains Telnet parameter statements that apply to all connections on all Telnet ports.

Use the following format in the PROFILE dataset:

```
►►─┬──────────────────────────────────────────────────────────────────┬───────►◄
   └─TELNETGLOBALS─┬─────────────────────────────────┬─ENDTELNETGLOBALS─┘
                   └─Telnet parameter ──statements─┘
```

TELNETPARMS statements

The TELNETPARMS block is a *required* statement block that contains Telnet parameter statements that apply to all connections of the Telnet port defined in the block. Use the following format in the PROFILE dataset:

```
►►──TELNETPARMS─┬─────────────────────────────────┬─ENDTELNETPARMS──────────►◄
                └─Telnet parameter ──statements─┘
```

PARMSGROUP statements

The PARMSGROUP Object statement is an *optional* statement that applies to connections which have the PARMSGROUP mapped by either their client identifiers or a matching LUMAP statement. Use the following format in the PROFILE dataset:

```
►►─┬───────────────────────────────────────────────────────────────────────┬──►◄
   └─PARMSGROUP──pg_name─┬─────────────────────────────────┬─ENDPARMSGROUP─┘
                         └─Telnet parameter ──statements─┘
```

BEGINVTAM block

The BEGINVTAM block is a *required* block that contains Telnet mapping statements used to *map objects to clients based on Client Identifier*. Use the following format in the PROFILE dataset:

```
►►──BEGINVTAM─┬───────────────────────────────┬─ENDVTAM───────────────────►◄
              └─Telnet mapping ──statements─┘
```

INCLUDE statement

The INCLUDE statement causes profile statements from the named data set to be included at the point that the INCLUDE statement is encountered.

Use the following format in the PROFILE dataset:

```
►►──INCLude──data_set_name──────────────────────────────────────────────────►◄
```

Telnet statement syntax

The statement syntax is the same in the configuration data set specified on the PROFILE DD card and the VARY TCPIP,tnproc,OBEYFILE command data set.

- A TELNETPARMS block and a BEGINVTAM block are required for each port.
- If a duplicate TELNETGLOBALS, TELNETPARMS for a port, BEGINVTAM for a port blocks, or PARMSGROUP name within a BEGINVTAM parameter is specified, the last statement block is used.
- If duplicate statements appear in the TELNETGLOBALS, TELNETPARMS, PARMSGROUP, or BEGINVTAM blocks, Telnet uses the last valid statement that was specified. However, if the REPLACEMENT statement is not valid, the statement being replaced is removed and replacement does not occur. This is referred to as the last one wins rule. The only exception to the last one wins rule is in the case of Client Identifiers defined in their respective group statement. For details, see “Telnet mapping statements in the Telnet profile” on page 635.
- Do not use the name of a Profile statement or parameter as a variable name in a statement. For example, do not assign the names USSTCP to USS table. Do not use the value GENERIC as a PRTGROUP name.
- For update capability and procedures, see z/OS Communications Server: IP Configuration Guide for information about managing Telnet.
- An END statement terminates a number of statements, such as the LUGROUP statement. If the END statement is omitted, all subsequent tokens in the data set are interpreted as parameters for that configuration statement until another statement is found.
- In general, if a syntax error is encountered in a list of parameters, such as an LUGROUP list, the parameter in error is ignored and the remaining entries are processed.
- Profile statements have some order restrictions. Basically, any statement that references a that is name defined in another statement must follow that statement. For example, LUMAP statements must follow the IPGROUP statement that defines the IPGROUP statement that is referenced by the mapping.
- During configuration, Telnet ensures that names are the appropriate length. If a name is too long, Telnet issues a message and the statement fails.
- Error messages are issued for incorrect statements. A DEBUG message displays the profile line number of the statement in error and other pertinent information. Error messages can be turned off by coding DEBUG OFF or DEBUG SUMMARY in the TELNETGLOBALS statement.
- A semicolon begins a comment. Comments act as blanks, separating words without affecting their meaning.
- An argument followed by a comment must have a blank before the semicolon.
- Statements can be split across multiple lines.
- Sequence numbers are not allowed.

Rules: User-defined names on configuration statements must adhere to the following rules:

- Entries in a configuration data set are free format; blanks, comments, and end-of-record are ignored.
- A configuration statement consists of a statement name followed by a required blank, and usually one or more positional arguments. Separate each argument by one or more blanks or end-of-records.
- Lowercase letters are translated to uppercase letters before the statements are executed, except for those parameters that support mixed case entries. For example, the KEYRING name is case sensitive.
- Static system symbols can be used in profile statements.
- Any IP address reference can be either an IPv4 format or IPv6 format IP address when the stack is running in IPv6-enabled mode.
- Each character must be a non-blank printable character.
- All characters must be entered in code page IBM-1047. The following are considered printable characters:

Table 31. Printable characters

Character	EBCDIC	Description
a-z	81-89, 91-99, A2-A9	Lowercase alphabetic
A-Z	C1-C9, D1-D9, E2-E9	Uppercase alphabetic
0-9	F0-F9	Numeric
¢	4A	Cent symbol
.	4B	Period
<	4C	Less than
(	4D	Left parenthesis
+	4E	Plus
|	4F	Vertical bar
&	50	Ampersand
!	5A	Exclamation
$	5B	Dollar
*	5C	Asterisk
)	5D	Right parenthesis
;	5E	Semicolon
^	5F	Hat
-	60	Minus, hyphen
/	61	Slash
,	6B	Comma
%	6C	Percent
_	6D	Underscore
>	6E	Greater than
?	6F	Question mark
`	79	Grave
:	7A	Colon
#	7B	Pound

Table 31. Printable characters (continued)

Character	EBCDIC	Description
@	7C	At
'	7D	Apostrophe
=	7E	Equal
"	7F	Double quote
~	A1	Tilda
[	AD	Left bracket
¬	BO	Logical not
]	BD	Right bracket
}	CO	Left brace
}	DO	Right brace
\	EO	Backslash

- The following printable characters cannot be used for many names. See specific statements for details.

Table 32. Restricted printable characters

Character	EBCDIC	Description
.	4B	Period
*	5C	Asterisk
;	5E	Semicolon
,	6B	Comma
=	7E	Equal

Telnet parameter statements in the Telnet profile

Table 33 provides a list of Telnet parameter statements and the location of more information.

The letter Y (with note references in parentheses) in a column indicates that the parameter can be coded in the indicated block. For example, CLIENTAUTH can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP (affecting all connections on all ports, all connections on one port, or a subset of connections on one port, respectively).

Table 33. Telnet parameter statements

Statement	TELNET GLOBALS	TELNET PARMS	PARMS GROUP	See
BINARYLINEMODE NOBINARYLINEMODE	Y	Y	Y See note 1.	"BINARYLINEMODE statement" on page 594
CHECKCLIENTCONN NOCHECKCLIENTCONN	Y	Y	Y See note 3.	"CHECKCLIENTCONN statement" on page 594
CLIENTAUTH	Y	Y	Y See note 2.	"CLIENTAUTH statement" on page 595
CODEPAGE	Y	Y	Y See note 1.	"CODEPAGE statement" on page 596

Table 33. Telnet parameter statements (continued)

Statement	TELNET GLOBALS	TELNET PARMS	PARMS GROUP	See
CONNTYPE		Y	Y See note 2.	“CONNTYPE statement” on page 597
CRLLDAPSERVER	Y			“CRLLDAPSERVER statement” on page 598
DBCSTRACE NODBCSTRACE		Y	Y	“DBCSTRACE statement” on page 599
DBCSTRANSFORM NODBSCTRANSFORM		Y	Y See note 3.	“DBCSTRANSFORM statement” on page 599
DEBUG	Y	Y	Y	“DEBUG statement” on page 600
DISABLEGSA	Y	Y	Y	“DISABLESGA statement” on page 602
DROPASSOCPRINTER NODROPASSOCPRINTER	Y	Y	Y	“DROPASSOCPRINTER statement” on page 602
ENCRYPTION	Y	Y	Y See note 2.	“ENCRYPTION statement” on page 602
EXPRESSLOGON NOEXPRESSLOGON	Y	Y	Y See note 3.	“EXPRESSLOGON statement” on page 603
FORMAT	Y			“FORMAT statement” on page 604
FULLDATATRACE NOFULLDATATRACE	Y	Y	Y	“FULLDATATRACE statement” on page 605
INACTIVE	Y	Y	Y	“INACTIVE statement” on page 605
KEEPINACTIVE	Y	Y	Y	“KEEPINACTIVE statement” on page 606
KEEPLU	Y	Y	Y	“KEEPLU statement” on page 607
KEYRING	Y	Y		“KEYRING statement” on page 607
LUSESSIONPEND NOLUSESSIONPEND	Y	Y	Y	“LUSESSIONPEND statement” on page 609
MAXRECEIVE	Y	Y	Y	“MAXRECEIVE statement” on page 609
MAXREQSESS	Y	Y	Y	“MAXREQSESS statement” on page 610
MAXRUCHAIN	Y	Y	Y	“MAXRUCHAIN statement” on page 610
MAXTCPSENDQ	Y	Y	Y	“MAXTCPSENDQ statement” on page 611
MAXVTAMSENDQ	Y	Y	Y	“MAXVTAMSENDQ statement” on page 611
MSG07 NOMSG07	Y	Y	Y See note 1.	“MSG07 statement” on page 612
NACUSERID NONACUSERID	Y	Y		“NACUSERID statement” on page 612

Table 33. Telnet parameter statements (continued)

Statement	TELNET GLOBALS	TELNET PARMS	PARMS GROUP	See
OLDSOLICITOR NOOLDSOLICITOR	Y	Y	Y	"OLDSOLICITOR statement" on page 613
PASSWORDPHRASE NOPASSWORDPHRASE DISABLEPASSWORDPHRASE	Y	Y	Y	"PASSWORDPHRASE statement" on page 613
PORT SECUREPORT TTLSPORT		Y		"PORT, SECUREPORT, and TTLSPORT statements" on page 614
PRTINACTIVE	Y	Y	Y	"PRTINACTIVE statement" on page 616
PROFILEINACTIVE	Y	Y	Y	"PROFILEINACTIVE statement" on page 615
REFRESHMSG10 NOREFRESHMSG10	Y	Y	Y	"REFRESHMSG10 statement" on page 616
SCANINTERVAL TIMEMARK	Y	Y	Y	"SCANINTERVAL and TIMEMARK statements" on page 617
SEQUENTIALLU NOSEQUENTIALLU	Y	Y	Y	"SEQUENTIALLU statement" on page 617
SGA NOSGA (DISABLESGA)	Y	Y	Y See note 3.	"SGA statement" on page 618
SHAREACB NOSHAREACB	Y			"SHAREACB statement" on page 618
SIMCLIENTLU NOSIMCLIENTLU	Y	Y	Y See note 3.	"SIMCLIENTLU statement" on page 619
SINGLEATTN NOSINGLEATTN	Y	Y	Y	"SINGLEATTN statement" on page 619
SMFINIT SMFTERM	Y	Y	Y	"SMFINIT and SMFTERM statements" on page 620
SMFPROFILE NOSMFPROFILE	Y			"SMFPROFILE statement" on page 621
SNAEXT NOSNAEXT	Y	Y	Y	"SNAEXT statement" on page 622
SSLTIMEOUT	Y	Y	Y See note 2.	"SSLTIMEOUT statement" on page 622
SSLV2 NOSSLV2	Y	Y	Y	"SSLV2 and NOSSLV2 statements" on page 623
SSLV3 NOSSLV3	Y	Y	Y	"SSLV3 and NOSSLV3 statements" on page 623
TCPIPJOBNAME NOTCPIPJOBNAME	Y			"TCPIPJOBNAME statement" on page 624
TELNETDEVICE	Y	Y	Y	"TELNETDEVICE statement" on page 624
TESTMODE		Y		"TESTMODE statement" on page 626
TIMEMARK	Y	Y	Y	"TIMEMARK statement" on page 626

Table 33. Telnet parameter statements (continued)

Statement	TELNET GLOBALS	TELNET PARMS	PARMS GROUP	See
TKOGENLU TKOGENLURECON NOTKO	Y	Y	Y	"TKOGENLU, TKOGENLURECON, and NOTKO statements" on page 627
TKOSPECLU TKOSPECLURECON NOTKO	Y	Y	Y	"TKOSPECLU, TKOSPECLURECON, and NOTKO statements" on page 628
TN3270E NOTN3270E	Y	Y	Y See note 3.	"TN3270E statement" on page 630
TNSACONFIG	Y			"TNSACONFIG statement" on page 631
UNLOCKKEYBOARD	Y	Y	Y	"UNLOCKKEYBOARD statement" on page 632
XCFGROUP	Y			"XCFGROUP statement" on page 633

Note:

1. Changing or setting the function at LU assignment time using the LUMAP-PMAP statement might not provide the expected results. Use PARMSMAP for consistent results.
2. The statement definition is used before the user ID Client Identifier is determined and before LU assignment is performed. To use the statement in PARMSGROUP, the group must be mapped using PARMSMAP to any Client Identifier other than user ID or user group.
3. The function is negotiated with the client before LU assignment; therefore, LUMAP PMAP has no affect on these statements.

Rules for Telnet parameter statements and security parameters

Observe the following rules for parameter statements:

- The value of parameter statements used by a connection is determined by the parameter hierarchy. All parameter values are initially set to Telnet default values and can then be modified using the TELNETGLOBALS block, TELNETPARMS block, or PARMSGROUP object. TELNETGLOBALS parameters affect all connections on all ports, TELNETPARMS parameters affect all connections on a single port, and PARMSGROUP parameters affect a subset of connections within a single port.
- If no statements are entered between TELNETPARMS and ENDTELNETPARMS, Telnet uses the default values for each of the TELNETPARMS statements.

Specific rules apply to security statements.

The CONNTYPE parameter statement is valid on a port with the value of SECUREPORT or TTLSPORT specified but not on a basic port.

Restrictions: The following parameter statements are valid on a SECUREPORT port only. The equivalent function for a TTLSPORT port is defined in AT-TLS policy.

- CRLLDAPSERVER
- CLIENTAUTH

- KEYRING
- ENCRYPTION
- SSLV2/NOSSLV2
- SSLV3/NOSSLV3

If any of these parameters are coded on or sift down to a basic port, or if any of these parameters are coded on or sift down to a TTLSPORT port, they are handled in the following ways:

- Security parameters in the TELNETGLOBALS block that sift down to basic or TTLSPORT ports are ignored, and a DEBUG warning message is issued. To avoid the DEBUG message, code the Security parameters in the TELNETPARMS block instead of in the TELNETGLOBALS block.
- Security parameters in the TELNETPARMS block for a basic or TTLSPORT port cause the port update to fail, and a DEBUG error message is issued. It is assumed that the port was meant to be a SECUREPORT port because of the presence of these parameters. Either specify the SECUREPORT parameter or remove the security parameters.
- Security parameters in the PARMSGROUP Object statement mapped in a basic or TTLSPORT port are ignored, and a DEBUG warning message is issued. To avoid the DEBUG messages, remove these parameters. If the BEGINVTAM block supports multiple ports (basic, TTLSPORT, and SECUREPORT), duplicate the BEGINVTAM block into multiple blocks and remove the security parameters from the basic or TTLSPORT port.

BINARYLINEMODE statement

The BINARYLINEMODE parameter statement is used to prohibit translation of characters between EBCDIC and ASCII during linemode sessions. If NOBINARYLINEMODE is specified, standard linemode translation is implemented.

Syntax

```
►►──┬──────────────────┬──────────────────────────────────────────────►◄
    ├─NOBINARYLINEMODE─┤
    └─BINARYLINEMODE───┘
```

Parameters

This statement has no parameters.

Telnet is initialized with the value NOBINARYLINEMODE.

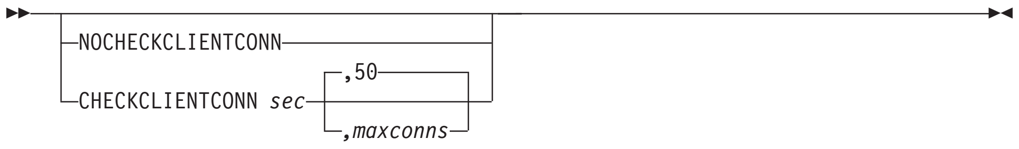
BINARYLINEMODE and NOBINARYLINEMODE can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

CHECKCLIENTCONN statement

Use the CHECKCLIENTCONN parameter statement to trigger the checking of the connectivity of all pre-existing connections associated with the client identifier of the new connection being established. The new connection is delayed early during connection negotiation until either all existing connections have responded or the specified wait time has elapsed. The number of existing connections checked can be limited with the *maxconns* parameter.

Guideline: No specific order is used when a limited number of connections are checked.

Syntax



Parameters

maxconns
: The maximum number of connections checked for a single client identifer. The connections are not checked in any particular order. The range is 1 - 99999999. The default value for *maxconns* is 50.

  Tip: This parameter can be important if you are using a proxy server. A proxy server causes all client connections to appear as if they are coming from the same client IP address. If you have a large number of connections coming in through a proxy server, Telnet sends timemarks out to each existing connection every time a new connection is established. The proxy server can be managed in either of the follow ways:

  - Use the Parmsgroup/Parmsmap statements to specify the NoCheckClientConn option for the proxy server.
  - Specify a small *maxconns* value to keep the number of connections checked for the proxy server low.

  Telnet is initialized with the value NOCHECKCLIENTCONN.

  The CHECKCLIENTCONN and NOCHECKCLIENTCONN statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

sec
: Number of seconds Telnet waits before checking whether a response was received from the client connections. Valid values are in the range 1 - 99 999 999.

CLIENTAUTH statement

Use the CLIENTAUTH parameter statement to specify whether or not client authentication is used for the SECUREPORT port.

Telnet is initialized with the value CLIENTAUTH NONE.

CLIENTUATH can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

CLIENTAUTH is valid only with a secure port. See “Rules for Telnet parameter statements and security parameters” on page 593 for details.

Syntax

```
►►──┬─────────────────────────────┬──────────────────────────────────────►◄
    └─CLIENTAUTH──┬─SSLCERT─┬─────┘
                  ├─SAFCERT─┤
                  └─NONE────┘
```

Parameters

SSLCERT
Specifies that the SSL handshake process authenticates the client certificate as well as the server certificate. This check verifies that the client has received a certificate from a trusted certificate authority (CA).

SAFCERT
Specifies that the SSL handshake process authenticates the client certificate. Prior to completing connection negotiation, additional access control is provided through the installation's SAF compliant security product (for example, RACF) as follows:

- Verifies that the client certificate has an associated user ID defined to the security product. The certificate must first be defined to the security product to obtain this validation. For more information about adding certificates to RACF, see the description of the RACDCERT command in the z/OS Security Server RACF Command Language Reference.
- For security products that support the 'SERVAUTH' class, installations can also obtain a more granular level of access control. If the installation has activated the SERVAUTH class and provided a profile for the port in the 'SERVAUTH' class, only users specified in the profile are allowed to connect into the port. See z/OS Communications Server: IP Configuration Guide for more information. The security product profile name is specified in the following format:

  ```
  EZB.TN3270.sysname.tcpname.PORTnnnnn
  ```

NONE
No client authentication checks are to be done.

CODEPAGE statement

Use the CODEPAGE parameter statement to specify ASCII-EBCDIC translation tables for linemode connections.

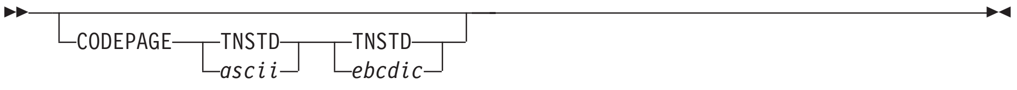
Telnet is initialized to use the ISO859-1 code page for ASCII and the IBM-1047 code page for EBCDIC.

CODEPAGE can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values. If there is an error in the syntax, a default code page of ISO8859-1 is used for ASCII and the language environment code page taken from locale information is used as the EBCDIC code page. If the EBCDIC code page is in error, a default code page of IBM-1047 is used for EBCDIC.

If TNSTD is specified as either parameter, TNSTD is used for both. The Telnet table is based on the ISO08859-1/IBM-1047 translation tables with the following exceptions:

```
EBCDIC          ASCII
x'0D25' -----> x'0D0085' using ISO8859-1/IBM-1047
x'0D25' -----> x'0D0A'   using internal tables
x'15'   <----  x'0A'     using ISO8859-1/IBM-1047
x'25'   <----  x'0A'     using internal tables
```

Syntax



Parameters

ascii
The ASCII code page name. If TNSTD is specified, the TELNET-created translation table is used.

ebcdic
The EBCDIC code page name. If TNSTD is specified, the TELNET-created translation table is used.

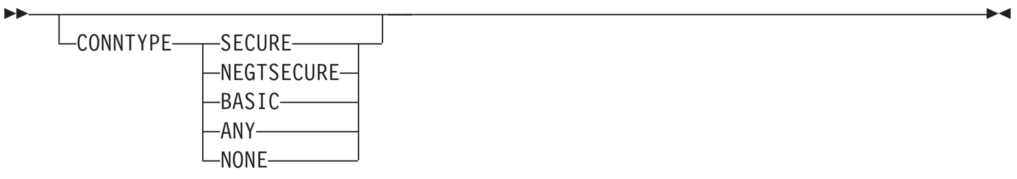
CONNTYPE statement

Use the CONNTYPE parameter statement select different connection types.

CONNTYPE can be coded in TELNETPARMS or PARMSGROUP. See "Rules for Telnet parameter statements and security parameters" on page 593 for parameter statements for more information about the hierarchy of parameter values.

CONNTYPE is valid only with a secure port. See "Rules for Telnet parameter statements and security parameters" on page 593 for details.

Syntax



Parameters

SECURE
Indicates that the traditional SSL handshake is used to start the SSL connection. If the client does not start the handshake within the time specified by SSLTIMEOUT, an attempt is made to do a negotiated SSL handshake. If the client rejects the negotiated attempt, the connection is closed.

Telnet is initialized for secure ports SECUREPORT or TTLSPORT with CONNTYPE SECURE and for basic ports with CONNTYPE BASIC.

NEGTSECURE
Indicates that a TN3270 negotiation with the client determines if the client is willing to enter into a secure connection. If the client agrees, SSL protocols are used for all subsequent communication. If the client does not agree, the connection is closed.

BASIC
Indicates that a basic (non-SSL) connection is used.

ANY
Indicates that the client can connect as secure or basic. Telnet first tries a standard SSL handshake. If the handshake times out, negotiated SSL (see CONNTYPE NEGTSECURE) is attempted.

- If the client is willing to enter into a secure connection, SSL protocols are used for all subsequent communication.
- If the client is not willing to enter into a secure connection, a basic connection is used.

NONE
Indicates that any client connection request is rejected.

CRLLDAPSERVER statement

Use the CRLLDAPSERVER parameter statement to specify the LDAP server or servers to be used for Telnet's Certificate Revocation List (CRL) processing. CRL processing using the LDAP server is done in conjunction with Telnet's SSL client authentication of client certificates. If the client's certificate is found on the certificate revocation list, the connection is closed. The anonymous user ID is used to connect to the CRLLDAPSERVER.

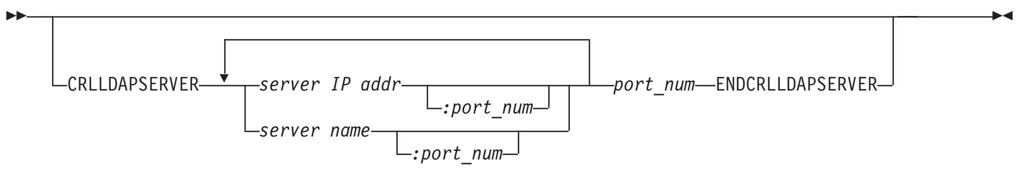
Restrictions:

- This statement does not support the IPv6 format.
- A maximum of five server names can be specified for a total length of 255 characters including blank separators. If specified, the optional *:port_num* value overrides the required port number specified. There must be no space between the server name and the *:port_num*.
- The CRLLDAPSERVER statement can be coded only in TELNETGLOBALS.
- The CRLLDAPSERVER statement is valid only with a SECUREPORT port. See "Rules for Telnet parameter statements and security parameters" on page 593 for details.

If a SECUREPORT port is active during a profile update, the CRLLDAPSERVER parameters cannot be changed. If a change is attempted, an error message is issued and the profile updates for the SECUREPORT port are rejected. To change CRLLDAPSERVER parameters, all SECUREPORT ports must be stopped first.

If all SECUREPORTs are stopped when a profile update occurs, the CRLLDAPSERVER is refreshed if a new SECUREPORT is activated.

Syntax



Parameters

server IP addr
The IP address of the CRL LDAP server.

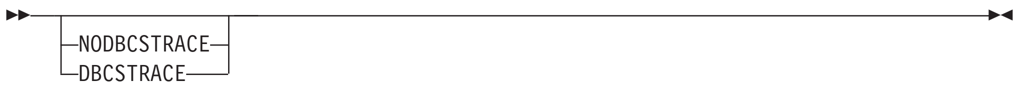
server name
The name of the CRL LDAP server.

port_num
The port number of the CRL LDAP server.

DBCSTRACE statement

Use the DBCSTRACE parameter statement to activate additional, detailed tracing within the DBCS load module. The trace records are written to the SYSPRINT and TNDBCSER file. If NODBCSTRACE is specified, detailed trace records are not written.

Syntax



Parameters

This statement has no parameters.

Telnet is initialized with the value NODBCSTRACE. DBCSTRACE and NODBCSTRACE can be coded in TELNETPARMS or PARMSGROUP. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

DBCSTRANSFORM statement

Use the DBCSTRANSFORM parameter statement to configure Telnet linemode to support 3270 SBCS or DBCS ASCII-EBCDIC transformations. The DBCSTRANSFORM statement specifies that Telnet should load the 3270 DBCS transform module, TNDBCSTM, at initialization. If the NODBCSTRANSFORM statement is specified, standard linemode translation is performed.

Telnet is initialized with the value NODBCSTRANSFORM. DBCSTRANSFORM and NODBCSTRANSFORM can be coded in TELNETPARMS or PARMSGROUP. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

The TNDBCSTM module must be in a data set in the system search list. You can find the module in the installation data set, SEZALOAD. If you are using the 3270 DBCS transform mode, the TCP/IP address space might require additional virtual storage. The TNDBCSCN, TNDBCSXL, and TNDBCSER DD statements must be provided in the started procedure's JCL when DBCSTRANSFORM is specified. See the linemode operation information in z/OS Communications Server: IP Configuration Guide for details about their usage.

Transform is supported only on a single port. To use transform on a different port, the port using transform must be stopped using VARY TCPIP,,T,STOP. Then an OBEYFILE command can be used to process a new Telnet profile, which defines transform support on another port.

If DBCSTRANSFORM is coded in multiple parameter blocks, the last port identified as DBCSTRANSFORM is the DBCSTRANSFORM port. The maximum number of transform connections is 250.

Syntax

```
►►──┬─────────────────┬──────────────────────────────────────────────────►◄
    ├─NODBCSTRANSFORM─┤
    └─DBCSTRANSFORM───┘
```

Parameters

This statement has no parameters.

DEBUG statement

Use the DEBUG parameter statement to provide different levels of debug information for Telnet problems or tracking. Without this statement, only certain connection drop reasons are reported to the operator console.

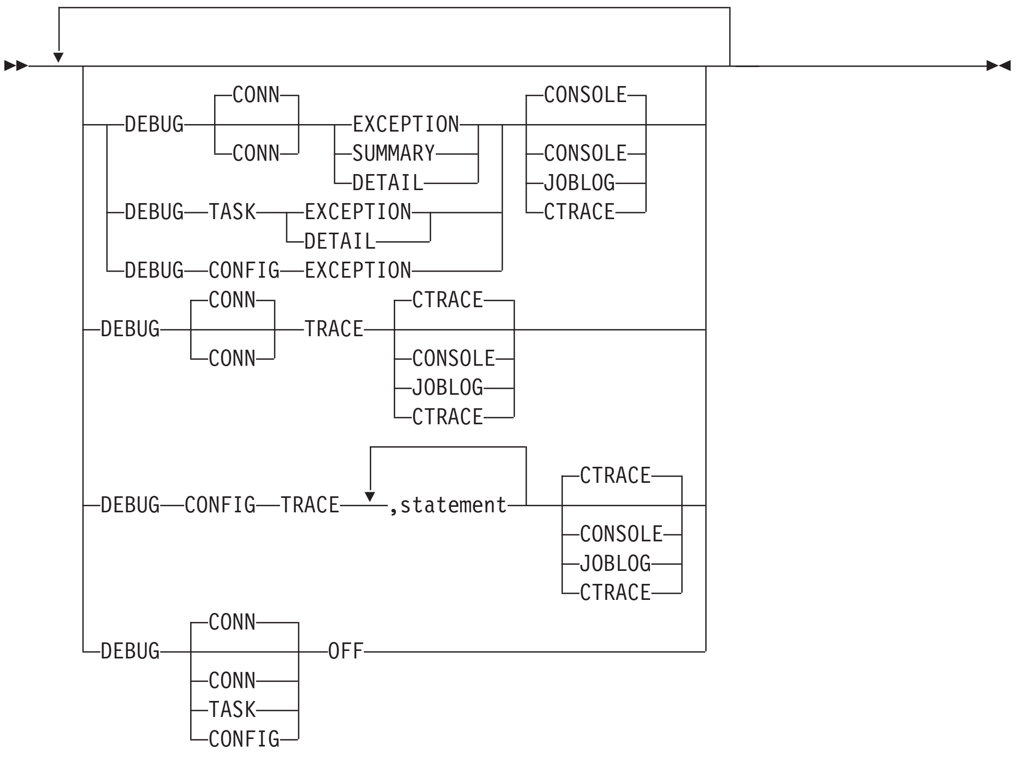
Telnet is initialized with DEBUG CONN EXCEPTION, DEBUG TASK EXCEPTION, and DEBUG CONFIG EXCEPTION settings.

You can code DEBUG CONN in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values. You can code DEBUG CONFIG and DEBUG TASK only in TELNETGLOBALS.

DEBUG CONN, DEBUG TASK, and DEBUG CONFIG can each be specified once in TelenetGlobals without generating a duplicate statement exception. The parameters EXCEPTION, SUMMARY, DEBUG, and TRACE are mutually exclusive on each of the three types of debug statements.

Use the V TCPIP,,T,DEBUG,OFF command to turn off all active debug reporting. This command also turns off the reporting of connection drops that were caused by timeouts or errors.

Syntax



Parameters

CONN
: Specify CONN to issue debug messages for connections. CONN is the default.

TASK
: Specify TASK to issue debug messages for Telnet tasks.

CONFIG
: Specify CONFIG to issue debug messages for Telnet configuration statements. A CONFIG debug message (EZZ6035I) is issued showing the statements and parameters read by Telnet. Another EZZ6035I message is issued showing the structure of the statement as it is passed to Telnet database processing.

OFF
: When OFF is specified, no debug messages are issued.

EXCEPTION
: When EXCEPTION is specified, only exception debug messages are issued. Telnet is initialized with the value DEBUG CONN EXCEPTION.

SUMMARY
: When SUMMARY is specified, summary debug messages (EZZ6034I) are issued indicating major state changes. EXCEPTION debug messages are also issued when SUMMARY is specified.

DETAIL
: When DETAIL is specified, detail debug messages (EZZ6035I) are issued to show key events occurring. You should specify DETAIL when you are solving problems; otherwise, too many messages are generated. EXCEPTION and SUMMARY messages are also issued when DETAIL is specified.

TRACE
: When TRACE is specified, data to and from the client and to and from VTAM

for one connection is displayed by debug message EZZ6035I. Detail and summary messages are also issued when TRACE is specified. When DEBUG CONFIG is specified, you can optionally specify statement names immediately after the TRACE parameter to indicate that only those statements should be displayed. You can specify a maximum of 20 statement names.

JOBLOG
When JOBLOG is specified, the debug messages are routed to the joblog (routing code 11) instead of the console.

CONSOLE
When CONSOLE is specified, the debug messages are routed to the master console (routing code 2) and to the teleprocessing console (routing code 8).

CTRACE
When CTRACE is specified, the debug messages are not issued and appear in the Component Trace only.

DISABLESGA statement

See the "SGA statement" on page 618 for information about this statement.

DROPASSOCPRINTER statement

Use the DROPASSOCPRINTER parameter statement to control whether or not the associated printer is dropped when the terminal connection is dropped.

Telnet is initialized with the value NODROPASSOCPRINTER.

DROPASSOCPRINTER and NODROPASSOCPRINTER can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬────────────────────┬──────────────────────────────────────────────►◄
    ├─NODROPASSOCPRINTER─┤
    └─DROPASSOCPRINTER───┘
```

Parameters

This statement has no parameters.

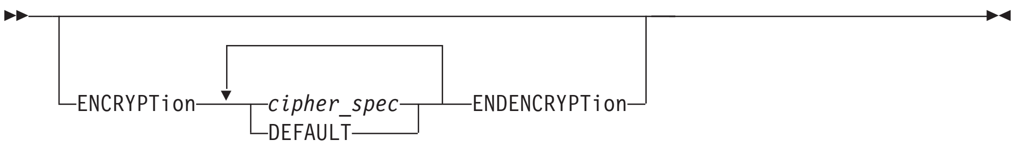
ENCRYPTION statement

Use the ENCRYPTION parameter statement to allow the selection of a subset of the supported algorithms to use for this port. Each z/OS system level supports a specific set of encryption algorithms.

The ENCRYPTION statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Restriction: The ENCRYPTION/ENDENCRYPTION block applies only to a Telnet SECUREPORT that serves SSLv3/TLSv1 and later clients.

Syntax



Parameters

cipher_spec
: The cipher specification (*cipher_spec*) to use for this port. The order in which the cipher specifications are specified is significant. The server controls which of the available cipher specifications are used for data encryption by specifying the desired cipher specification in order of preference. The actual *cipher_spec* used is the best match between what the server requests and what the client supports. If the client does not support any of the cipher specifications the server requests, the secure handshake fails and the connection is closed.

DEFAULT
: Indicates that the cipher specifications, in the order listed below, are used for SSLv3 and TLSv1 negotiated connections.

Following are the cipher specifications that can be specified:

cipher_spec	Telnet Display Abbreviation	Cipher number
SSL_RC4_SHA	4S	05
SSL_RC4_MD5	4M	04
SSL_AES_256_SHA	A2	35
SSL_AES_128_SHA	A1	2F
SSL_3DES_SHA	3S	0A
SSL_DES_SHA	DS	09
SSL_RC4_MD5_EX	4E	03
SSL_RC2_MD5_EX	2E	06
SSL_NULL_SHA	NS	02
SSL_NULL_MD5	NM	01
SSL_NULL_Null	NN	00

All SSLv2 cipher specifications supported by System SSL are used for SSLv2 negotiated connections. The DEFAULT keyword provides a way to override specific choices made in TELNETGLOBALS or TELNETPARMS statements. If the DEFAULT keyword is specified along with a *cipher_spec* value, only DEFAULT is recognized.

EXPRESSLOGON statement

Use the EXPRESSLOGON parameter statement to allow a user at a workstation, with a TELNET client and a X.509 certificate, to log on to an SNA application without entering a user ID or password. If NOEXPRESSLOGON is specified, EXPRESSLOGON function is not available to the client.

Telnet is initialized with the value NOEXPRESSLOGON.

The EXPRESSLOGON and NOEXPRESSLOGON statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See

"Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Requirements:

- The client must support the new environment Telnet option as defined in RFC 1572.
- When you are configuring the SECUREPORT value, you must specify CLIENTAUTH SAFCERT.
- When you are configuring the TTLSPORT value, the AT-TLS policy must specify HandshakeRole ServerWithClientAuth, a certificate must be received from the client, and the certificate must have an associated user ID.

Syntax

```
►►─────────────────────────────────────────────────────────────────────►◄
     ├─NOEXPRESSLOGON─┤
     └─EXPRESSLOGON───┘
```

Parameters

This statement has no parameters.

FORMAT statement

Use the FORMAT parameter statement to select the print format for display messages that are affected by longer IPv6 addresses.

Restriction: The FORMAT statement can be coded only in the TELNETGLOBALS statement block.

Syntax

```
►►──┬──────────────────────┬───────────────────────────────────────────►◄
    └─FORMAT──┬─SHORT─┬─┘
              └─LONG──┘
```

Parameters

SHORT
: The affected displays are presented in the existing one-line format. Telnet is initialized with a one-line format (FORMAT SHORT) in an IPv4 environment. A value of FORMAT SHORT cannot be coded in an IPv6 environment. All affected displays in an IPv6 environment use the new-wrapped line format.

LONG
: The affected displays are presented in a new format that accommodate IPv6 addresses. Even if the Client Identifier is short enough for the existing one-line format, the new two-line format is used. This parameter can be used as a migration tool to see the new two-line display formats without specifying an IPv6 environment. Telnet is initialized with a two-line format (FORMAT LONG) in an IPv6 environment or whenever an IPv6 address is specified in the profile.

FULLDATATRACE statement

Use the FULLDATATRACE parameter statement to specify that all data to and from the client and all data to and from VTAM is completely traced when the CTRACE, TELNET OPTION, is chosen. If NOFULLDATATRACE is specified, the first 64 bytes of data are traced.

Telnet is initialized with the value NOFULLDATATRACE.

The FULLDATATRACE and NOFULLDATATRACE statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─────────────────┬──────────────────────────────────────────────►◄
    ├─NOFULLDATATRACE─┤
    └─FULLDATATRACE───┘
```

Parameters

This statement has no parameters.

INACTIVE statement

Use the INACTIVE parameter statement to define the terminal SNA session inactivity timeout. A connection that has no client-VTAM session activity for the specified time is dropped.

Telnet is initialized with a INACTIVE value of 0.

The INACTIVE statement can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Restriction: The INACTIVE statement applies to a KEEPOPEN connection only when an SNA session, with the VTAM application, is active.

Telnet uses one timer for the INACTIVE, PRTINACTIVE, and KEEPINACTIVE statements. See z/OS Communications Server: IP Configuration Guide for details.

Syntax

```
►►──┬──────────────┬─────────────────────────────────────────────────►◄
    ├─INACTIVE 0───┤
    └─INACTIVE──sec┘
```

Parameters

0 An INACTIVE timeout value of 0 disables the inactivity timeout.

sec
: Sets the inactivity timeout to the specified number of seconds. When a connection has had no session activity for the specified number of seconds, it is closed. This number must be an integer in the range 0 - 99 999 999.

INCLUDE statement

This statement causes profile statements from the named data set to be included at the point that the INCLUDE statement is encountered. In general, a profile statement must begin and end within the same data set. For example, the statement beginning with BSDROUTINGPARMS and ending with ENDBSDROUTINGPARMS must be contained within the same data set. There are two exceptions to this requirement:

- INCLUDE statements can be used within the BEGINVTAM - ENDVTAM block of statements.
- INCLUDE statements can be used within a list of LUNAMES.

Syntax

```
►►──INCLude──data_set_name─────────────────────────────────────────────►◄
```

Parameters

data_set_name
: A fully qualified data set name that identifies a sequential file. The sequential file can be a sequential data set or a PDS with the member name. It cannot be a z/OS UNIX file.

KEEPINACTIVE statement

Use the KEEPINACTIVE parameter statement to define the session setup inactivity timeout. A KEEPOPEN connection with no active SNA session that has no client-VTAM activity for the specified time is dropped.

Telnet is initialized with a KEEPINACTIVE value of 0.

The KEEPINACTIVE statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Restriction: The KEEPINACTIVE statement applies to a KEEPOPEN connection only when the connection does not have an active SNA session.

Telnet uses one timer for the INACTIVE, PRTINACTIVE, and KEEPINACTIVE statements. See z/OS Communications Server: IP Configuration Guide for details.

Syntax

```
►►──┬─────────────────┬────────────────────────────────────────────────►◄
    ├─KEEPINACTIVE 0──┤
    └─KEEPINACTIVE──sec─┘
```

Parameters

0 A KEEPINACTIVE timeout of 0 disables the inactivity timeout.

sec
Sets the inactivity timeout to the specified number of seconds. When a KEEPOPEN connection has had no session for the specified number of seconds, it is closed. This number must be an integer in the range 0 - 99 999 999.

KEEPLU statement

Use the KEEPLU parameter statement to reserve the LU for the Client Identifier when the LU is unassigned from the connection. The first reconnection request from the same Client Identifier mimics an end user requesting a specific connection with the kept LU name.

Telnet is initialized with a KEEPLU value of 0.

The KEEPLU statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Restriction: The KeepLU function cannot be performed for specific connection requests or associated printer requests.

If both KEEPLU and SEQUENTIALLU statements are active, the KEEPLU value is used.

Syntax

```
►►──┬─────────────┬──────────────────────────────────────────────►◄
    ├─KEEPLU──0───┤
    └─KEEPLU──sec─┘
```

Parameters

0 A KEEPLU timeout of 0 disables the KEEPLU function.

sec
Sets the KEEPLU timeout to the specified number of seconds. When the LU has remained unassigned for the specified number of seconds, it becomes generally available. This number must be an integer in the range 0 - 99 999 999.

KEYRING statement

Use the KEYRING parameter statement to define the key ring to be used by Telnet SSL processing. This key ring contains the server certificate and keys to be used by Telnet and any CA Certificates required to do client authentication checks. If this statement is not coded, a secure port cannot be started.

The KEYRING statement can be coded in the TELNETGLOBALS or TELNETPARMS statement blocks. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

All uses of the KEYRING statement must specify the same data type and name. If coded in the TELNETGLOBALS statement block, any TELNETPARMS KEYRING values must match. If they do not, the port update is rejected. If KEYRING is not coded in TELNETGLOBALS but is coded on several TELNETPARMS, the last TELNETPARMS KEYRING value is assumed to be correct, and all other values must match it.

If all SECUREPORT ports (specified on the SECUREPORT statement) are stopped when a profile update occurs, the KEYRING file is refreshed when a new SECUREPORT port is activated.

Restrictions:

- The KEYRING statement is valid only with a SECUREPORT port. See "Rules for Telnet parameter statements and security parameters" on page 593 for details.
- If a SECUREPORT port is active during a profile update, the KEYRING name cannot change. If a change is attempted, an error message is issued for this parameter and the profile update for the related port is rejected. To change the KEYRING name, all SECUREPORT ports must first be stopped.

Syntax

```
►►─┬────────────────────────────────┬──────────────────────────────────►◄
   └─KEYRING───┬─HFS──hfsdsname─────┬┘
               ├─MVS──mvsdsname─────┤
               └─SAF──keyringname───┘
```

Parameters

HFS *hfsdsname*
: The path and file name of the key ring file. The name is case sensitive and can be any printable character except slash (/), which has an EBCDIC value of 61.

MVS *mvsdsname*
: The fully-qualified MVS data set name of the key ring that uses the RACF ADDRING function.

SAF *keyringname*
: The ring name specified when creating a key ring using the RACF ADDRING function. The name is case sensitive and can be any printable character except those shown in Table 34.

Table 34. SAF keyringname printable character exceptions

Character	EBCDIC	Description
(	4D	Left parenthesis
&	50	Ampersand
*	5C	Asterisk
)	5D	Right parenthesis
;	5E	Semicolon
,	6B	Comma

Table 34. SAF keyringname printable character exceptions (continued)

Character	EBCDIC	Description
%	6C	Percent
[	AD	Left bracket
¬	BO	Logical no
]	BD	Right bracket

LUSESSIONPEND statement

Use the LUSESSIONPEND parameter statement to enable Telnet to redrive the DEFAULTAPPL, USS, or Solicitor screen after LOGOFF of the current session. If the NOLUSESSIONPEND value is specified, the Telnet connection is dropped after session LOGOFF.

Telnet is initialized with a value of NOLUSESSIONPEND.

The LUSESSIONPEND and NOLUSESSIONPEND statements can be coded in TELNETGLOBALS, TELNETPARMS, and PARMSGROUP statement blocks. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►─┬─────────────────┬────────────────────────────────────────────►◄
   ├─NOLUSESSIONPEND─┤
   └─LUSESSIONPEND───┘
```

Parameters

This statement has no parameters.

MAXRECEIVE statement

Use the MAXRECEIVE parameter statement to limit the number of bytes received from a client without an End of Record (EOR) being received. If the amount of data received exceeds the limit, the connection is dropped. This parameter protects against a client in a send-data loop.

Telnet is initialized with a MAXRECEIVE value of 65 535.

The MAXRECEIVE statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values. A low value (less than 10 000) can cause unintended connection drops.

Syntax

```
►►─┬───────────────────────┬──────────────────────────────────────►◄
   ├─MAXRECEIVE ──0────────┤
   └─MAXRECEIVE ──num_bytes┘
```

Parameters

0 A MAXRECEIVE value of 0 disables the limit check function.

num_bytes
Sets the number of data bytes permitted to be received without receiving an EOR. This number must be an integer in the range 0 - 99 999 999.

MAXREQSESS statement

Use the MAXREQSESS parameter statement to limit the number of session requests received by Telnet in a 10-second period. For this parameter, a BIND received by Telnet defines a session request. If the number of BINDs received in a 10-second period exceeds the limit, the connection is dropped and an error is reported.

Telnet is initialized with a MAXREQSESS value of 20.

The MAXREQSESS statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬────────────────────────┬──────────────────────────────────────────►◄
    ├─MAXREQSESS ──0─────────┤
    └─MAXREQSESS ──num_req───┘
```

Parameters

0 A MAXREQSESS value of 0 disables the limit check function.

num_req
Sets the number of session requests permitted in a 10-second period. This number must be an integer in the range 0 - 99 999 999.

MAXRUCHAIN statement

Use the MAXRUCHAIN parameter statement to limit the number of chained RUs received from an application without an end of chain (EC) being received. If the number of RUs received exceeds the limit, the session, and conditionally the connection, is dropped. This parameter protects against a host application from sending too much chained data.

Telnet is initialized with a MAXRUCHAIN value of 0.

The MAXRUCHAIN statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─MAXRUCHAIN ──0───────┬─────────────────────────────────────────────►◄
    └─MAXRUCHAIN ──num_RUs─┘
```

Parameters

0 A MAXRUCHAIN value of 0 disables the function.

num_RUs
: Sets the number of chained RUs permitted to be received before the RU chain is ended. This number must be an integer in the range 0 - 99 999 999.

MAXTCPSENDQ statement

Use the MAXTCPSENDQ parameter statement to limit the number of bytes that are queued to be sent to a Telnet client. If the queue size exceeds the limit, the connection is dropped. This parameter prevents large amounts of storage from being held for data that is destined for an unresponsive Telnet client.

Telnet is initialized with a MAXTCPSENDQ value of 0.

The MAXTCPSENDQ statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─MAXTCPSENDQ ──0─────────┬──────────────────────────────────────────►◄
    └─MAXTCPSENDQ ──num_bytes─┘
```

Parameters

0 A MAXTCPSENDQ value of 0 disables the limit check function.

num_bytes
: Sets the number of bytes that can be queued to a Telnet client at one time. This number must be an integer in the range 0 - 99999999. A low value can cause unintended connection drops.

MAXVTAMSENDQ statement

Use the MAXVTAMSENDQ parameter statement to limit the number of data segments (RPLs) queued to be sent to VTAM. If the queue size exceeds the limit, the connection is dropped. This parameter protects against using large amounts of storage to contain data destined for a host VTAM application that is not receiving data.

Telnet is initialized with a MAXVTAMSENDQ value of 50.

The MAXVTAMSENDQ statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─────────────────────────┬──────────────────────────────────────►◄
    ├─MAXVTAMSENDQ ──0────────┤
    └─MAXVTAMSENDQ ──num_rpls─┘
```

Parameters

0 A MAXVTAMSENDQ value of 0 disables the limit check function.

num_rpls
: Sets the number of RPLs permitted to be queued to VTAM at one time. This number must be an integer in the range 0 - 99 999 999. A value less than 10 can cause unintended connection drops.

MSG07 statement

Use the MSG07 parameter statement to activate logon error message processing. Specifying this statement provides information to the client when a session attempt to the target application fails. If NOMSG07 is specified, the connection is dropped if a session initiation error occurs.

Telnet is initialized with a value of NOMSG07.

The MSG07 and NOMSG07 statements can be coded in TELNETGLOBALS, TELNETPARMS, and PARMSGROUP statement blocks. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─────────┬───────────────────────────────────────────────────────►◄
    ├─NOMSG07─┤
    └─MSG07───┘
```

Parameters

This statement has no parameters.

NACUSERID statement

Use the NACUSERID statement to associate one or more Telnet ports with a user ID defined to the security server. This provides Network Access Control checking with a user ID other than Telnet's address space user ID. If NONACUSERID is specified, Network Access Control uses Telnet's address space user ID.

Telnet is initialized with a value of NONACUSERID.

The NACUSERID and NONACUSERID statements can be coded in the TELNETGLOBALS or TELNETPARMS statement blocks.

Syntax

```
►►──┬──────────────────────┬──────────────────────────────────────────────►◄
    ├─NACUSERID NAC_name─┤
    └─NONACUSERID────────┘
```

Parameters

NAC_name
: Any valid user ID up to 8 characters in length.

OLDSOLICITOR statement

Use the OLDSOLICITOR parameter statement to place the initial cursor on the solicitor panel after the following prompt:

```
Enter Your Userid:
```

If NOOLDSOLICITOR is specified, the cursor is placed after the following prompt:

```
Application:
```

Telnet is initialized with a value of NOOLDSOLICITOR.

The OLDSOLICITOR and NOOLDSOLICITOR statements can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬──────────────────┬──────────────────────────────────────────────────►◄
    ├─NOOLDSOLICITOR─┤
    └─OLDSOLICITOR───┘
```

Parameters

This statement has no parameters.

PASSWORDPHRASE statement

Use the PASSWORDPHRASE statement to provide additional entry space on the solicitor screen for end users to enter either a password or a password phrase. When a new password or password phrase is entered, the user is asked to verify the new value.

Rule: The PASSWORDPHRASE statement is only effective if RESTRICTAPPL is coded. This operand affects only the solicitor screen and does not affect the password phrase capability of applications accessed using Telnet.

If NOPASSWORDPHRASE is specified, the solicitor screen continues to provide space for only a password.

If DISABLEPASSWORDPHRASE is specified, the solicitor screen provides support for only a password. The password is sent in the same format as it was in before z/OS V1R13.

Tip: The layout of the solicitor screen is changed to accommodate the larger password phrase. If you use a screen scraper, you must account for this change. If your client does not use 3270 data stream commands, code DISABLEPASSWORDPHRASE until the client is updated.

TELNET is initialized with the value NOPASSWORDPHRASE.

The DISABLEPASSWORDPHRASE, PASSWORDPHRASE and NOPASSWORDPHRASE statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─────────────────────┬──────────────────────────────────────────►◄
    ├─DISABLEPASSWORDPHRASE─┤
    ├─NOPASSWORDPHRASE──────┤
    └─PASSWORDPHRASE────────┘
```

Parameters

This statement has no parameters.

PORT, SECUREPORT, and TTLSPORT statements

Use the PORT parameter statement to define the port that Telnet listens on for non-secure (basic) connection requests.

Use the SECUREPORT parameter statement to define the port that Telnet listens on for secure connection requests from a client using the SSL protocol. If the SECUREPORT parameter statement is not coded, Telnet does not support secure access from a client.

Use the TTLSPORT parameter statement to define the port that Telnet listens on for secure connection requests from a client that uses the TCP/IP AT-TLS interface. If you use the TTLSPORT statement, then you can define security parameters in AT-TLS policy and not in the Telnet profile.

Telnet is initialized with a value of PORT 23.

Restrictions:
- You can code the PORT, SECUREPORT, or TTLSPORT statements only in the TELNETPARMS statement block.
- If you code a qualifier value (*qual*), it must match the qualifier used in the PORT statement in the BEGINVTAM block.

If you code SECUREPORT, you can also specify several Telnet security parameter statements; see “Rules for Telnet parameter statements and security parameters” on page 593 for details. Specifying SECUREPORT or TTLSPORT is the same as specifying CONNTYPE SECURE; PORT is the same as specifying CONNTYPE BASIC. See “CONNTYPE statement” on page 597 for more information.

In the BEGINVTAM block, the PORT statement serves a different purpose. It links the BEGINVTAM block to the TELNETPARMS block with the same port number.

Syntax

```
►►─┬─────────────────────────────┬─────────────────────────────────────►◄
   ├─PORT 23─────────────────────┤
   ├─PORT──────┬─num──────┬──────┤
   │           └─num,qual─┘      │
   ├─SECUREPORT──┬─num──────┬────┤
   │             └─num,qual─┘    │
   └─TTLSPORT──┬─num──────┬──────┘
               └─num,qual─┘
```

Parameters

num
: A specified port number.

,qual
: Qualifies the port address (PORT) with a destination IP address or with a specific link name.

PROFILEINACTIVE statement

Use the PROFILEINACTIVE parameter statement to define the timeout for connections associated with a non-current profile that do not have a SNA session. A connection that does not have a SNA session for the specified time and that is associated with a non-current profile is dropped. Telnet uses one timer for the INACTIVE, PROFILEINACTIVE, PRTINACTIVE, and KEEPINACTIVE statements. See z/OS Communications Server: IP Configuration Guide for more information. You can code the PROFILEINACTIVE statement in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Telnet is initialized with a PROFILEINACTIVE timeout value of 1800 seconds.

Syntax

```
►►─┬──────────────────────────┬──────────────────────────────────────►◄
   ├─PROFILEINACTIVE ──1800───┤
   └─PROFILEINACTIVE ──sec────┘
```

Parameters

1800
: This PROFILEINACTIVE timeout value sets the inactivity value to 1800 seconds.

sec
: Sets the inactivity timeout to the specified number of seconds. This number must be an integer in the range 0-99 999 999. A PROFILEINACTIVE timeout value of 0 disables the function.

PRTINACTIVE statement

Use the PRTINACTIVE parameter statement to define the printer inactivity timeout. A printer connection with no client-VTAM activity for the specified time is dropped.

Telnet is initialized with a PRTINACTIVE value of 0.

The PRTINACTIVE statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Telnet uses one timer for INACTIVE, PRTINACTIVE, and KEEPINACTIVE. See z/OS Communications Server: IP Configuration Guide for more details.

Syntax

```
►►──┬──────────────────┬──────────────────────────────────────────────►◄
    ├─PRTINACTIVE 0────┤
    └─PRTINACTIVE ──sec┘
```

Parameters

0 A PRTINACTIVE timeout value of 0 disables inactivity timeout.

sec
Sets the inactivity timeout to a specified number of seconds. When a printer connection has been inactive for the specified number of seconds, it is closed. The number must be an integer in the range 0 - 99 999 999.

REFRESHMSG10 statement

Use the REFRESHMSG10 parameter statement to specify the action to be taken when a clear key is entered from a USSMSG message. If the REFRESHMSG10 parameter statement is specified, Telnet toggles between clearing the screen and returning to the USSMSG10 panel. When the screen is cleared, the cursor is placed at location row one and column two.

If the NOREFRESHMSG10 parameter statement is specified, the screen is always cleared, and the cursor is placed at location row one and column one.

Telnet is initialized with the REFRESHMSG10 parameter statement.

The REFRESHMSG10 and NOREFRESHMSG10 parameter statements can be coded in the TELNETGLOBALS, TELNETPARMS and PARMSGROUP statement blocks.

Syntax

```
►►──┬────────────────┬────────────────────────────────────────────────►◄
    ├─REFRESHMSG10───┤
    └─NOREFRESHMSG10─┘
```

Parameters

This statement has no parameters.

SCANINTERVAL and TIMEMARK statements

Use the SCANINTERVAL parameter statement to define the interval at which Telnet checks connections for inbound TCP/IP activity. It is used in conjunction with the TIMEMARK parameter statement, which defines the elapsed time Telnet uses to determine whether a connection to the client is considered broken. During SCANINTERVAL processing, if the elapsed time since the last inbound activity is greater than the TIMEMARK value, the connection is considered possibly broken and a TIMEMARK request is sent to the client. At the next interval, if neither a TIMEMARK request nor data is received, the connection is considered broken. Telnet drops the connection.

SCANINTERVAL and TIMEMARK can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

If for any reason the TIMEMARK cannot be sent immediately on five consecutive tries and no data or TIMEMARK response is received, the connection is dropped. When TIMEMARK cannot be sent immediately, Telnet tries again at the next SCANINTERVAL time. If the SCANINTERVAL is greater than the TIMEMARK value, it is reset to the TIMEMARK value.

Syntax

```
►►─┬─────────────────────┬─┬───────────────────┬───────────────────────────────►◄
   ├─SCANINTERVAL 1800───┤ ├─TIMEMARK 10,800─┤
   └─SCANINTERVAL ──sec1─┘ └─TIMEMARK ──sec2─┘
```

Parameters

1800
Telnet is initialized with a SCANINTERVAL value of 1800 seconds.

sec1
Sets the SCANINTERVAL time to a specified number of seconds. This value is in the range 1 - 99 999 999. A value of 0 is not valid.

10,800
Telnet is initialized with a TIMEMARK value of 10 800 seconds.

sec2
Sets the TIMEMARK time to a specified number of seconds. This value is in the range 1 - 99 999 999. A value of 0 is not valid.

SEQUENTIALLU statement

Use the SEQUENTIALLU parameter statement allows sequential LU selection from the LU group. If NOSEQUENTIALLU is specified, the first LU available in the group is used.

Telnet is initialized with SEQUENTIALLU.

SEQUENTIALLU and NOSEQUENTIALLU can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬────────────────┬──────────────────────────────────────────────►◄
    ├─SEQUENTIALLU───┤
    └─NOSEQUENTIALLU─┘
```

Parameters

This statement has no parameters.

SGA statement

Use the NOSGA (DISABLESGA) parameter statement to permit the transmission of GO AHEAD by Telnet. It is negotiated by both client and server. Using NoSGA increases the overhead for a full duplex terminal and a full duplex connection. If SGA is specified, transmission of GO AHEAD is suppressed.

Telnet is initialized with a value of SGA.

The SGA and NOSGA statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬────────────────────┬──────────────────────────────────────────►◄
    ├─SGA────────────────┤
    └─NOSGA (DISABLESGA)─┘
```

Parameters

This statement has no parameters.

SHAREACB statement

Use the SHAREACB parameter statement to allow multiple Telnet LUs to share an ACB. Telnet server can use ECSA storage more efficiently by using this parameter. If NOSHAREACB is specified, ACB sharing does not occur.

Telnet is initialized with a value of NOSHAREACB.

Guideline: Use of the SHAREACB statement requires Telnet LUs to be defined with model APPL names in the VTAM configuration data set.

The SHAREACB and NOSHAREACB statements can be coded in the TELNETGLOBALS statement block only.

Syntax

```
►►──┬─NOSHAREACB─┬──────────────────────────────────────────────────►◄
    └─SHAREACB───┘
```

Parameters

This statement has no parameters.

SIMCLIENTLU statement

Use the SIMCLIENTLU parameter statement to cause Telnet to send a standard LU name (EZBSIMLU) during negotiation to any TN3270E client requesting a Generic connection. Instead of assigning a Telnet LU, the LU assignment is deferred until after application selection, just like TN3270 clients. If NOSIMCLIENTLU is specified, normal device name negotiation occurs for TN3270E connections.

Telnet is initialized with a value of NOSIMCLIENTLU.

The SIMCLIENTLU and NOSIMCLIENTLU statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

When the TN3270E client requests a connection with a specific LU, the selection of the LU is handled like normal TN3270E specific processing, regardless of the SIMCLIENTLU statement. Printer requests are not affected by the SIMCLIENTLU statement.

Syntax

```
►►──┬─NOSIMCLIENTLU─┬───────────────────────────────────────────────►◄
    └─SIMCLIENTLU───┘
```

Parameters

This statement has no parameters.

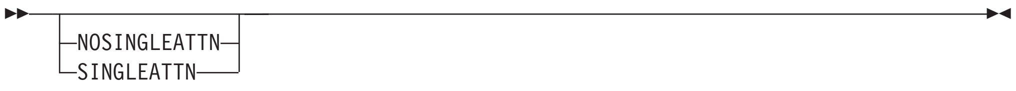
SINGLEATTN statement

Use the SINGLEATTN parameter statement to cause Telnet to check the data for a double ATTENTION key combination, x'6CFFEFFFF3', in the data stream sent from the client. If a double ATTENTION key combination is found, Telnet sends only a single ATTENTION. If NOSINGLEATTN is specified, the data is not checked.

Telnet is initialized with a value of NOSINGLEATTN.

The SINGLEATTN and NOSINGLEATTN statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax



Parameters

This statement has no parameters.

SMFINIT and SMFTERM statements

Use the SMFINIT and SMFTERM parameter statements to configure Telnet to write SMF records. These statements control the invocation of Telnet SNA Session Initiation (or LOGON, subtype 20) and Telnet SNA Session Termination (or LOGOFF, subtype 21) SMF records.

Two different record formats are available:

- Format 118
- Format 119

The format 119 records are controlled by use of the TYPE119 operand on the SMFINIT and SMFTERM statements. The specification of the STD operand or a nonstandard subtype number on the SMFINIT and SMFTERM statements control the usage of the older format 118 record processing.

Telnet is initialized with the following values:

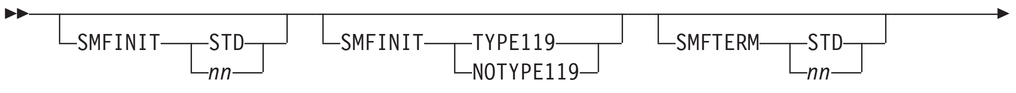
- SMFINIT 0
- SMFINIT NOTYPE119
- SMFTERM 0
- SMFTERM NOTYPE119

SMFINIT and SMFTERM can be coded in TELNETGLOBALS, TELNETPARMS, and PARMSGROUP. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

TCP/IP SMF records are independent of the IP connection. They are created for Telnet LU/HOST application sessions.

Many products use standard SMF record subtypes. A standard subtype avoids potential double usage and makes it easier for other vendors to write SMF output processing programming and for Telnet administrators to be consistent across multiple machines.

Syntax



```
►─┬───────────────────────┬──────────────────────────────────────────────►◄
  └─SMFTERM──┬─TYPE119───┤
             └─NOTYPE119─┘
```

Parameters

STD
: Specifies that format 118 SMF records should be written using standard subtypes for LOGON (20) or LOGOFF (21) records.

nn
: Specifies the format 118 SMF record subtype for LOGON or LOGOFF records. Valid values are integers in the range 0 - 255. A value of 0 for SMFINIT and SMFTERM indicates that no SMF record is written for that function. The user can change the subtype value only for the format 118 records.

TYPE119
: Specifies that format 119 SMF records should be written for Telnet SNA Session Initiation (subtype 20) or Telnet SNA Session Termination (subtype 21) records.

NOTYPE119
: Specifies that format 119 SMF records should not be written.

SMFPROFILE statement

Use the SMFPROFILE parameter statement to configure Telnet to write SMF configuration records. The Telnet configuration records are written as type 119, subtype 24. For more information about the layout of the configuration records, see z/OS Communications Server: IP Configuration Guide.

Telnet is initialized with a value of NOSMFPROFILE.

The SMFPROFILE and NOSMFPROFILE statements can be coded in only the TELNETGLOBALS statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►─┬──────────────────────────────────────────────┬──────────────────────►◄
   ├─NOSMFPROFILE─────────────────────────────────┤
   │             ┌─GROUPDETAIL──────────────┐     │
   └─SMFPROFILE──┴─┬────────────────────────┬┴────┘
                   └─┬─NOGROUPDETAIL─┬──────┘
                     └─GROUPDETAIL───┘
```

Parameters

GROUPDETAIL | NOGROUPDETAIL
: GROUPDETAIL includes all of the individual members of the Telnet group statements in the SMF records. NOGROUPDETAIL includes only the first 10 members of each group. GROUPDETAIL is the default. The following list shows Telnet group statements:

- ALLOWAPPL or RESTRICTAPPL
- DEFAULTLUS or SDEFAULTLUS
- DEFAULTLUSSPEC or SDEFAULTLUSSPEC
- DEFAULTPRT or SDEFAULTPRT

- DEFAULTPRTSPEC or SDEFAULTPRTSPEC
- DESTIPGROUP
- HNGROUP
- IPGROUP
- LINKGROUP
- LUGROUP or SLUGROUP
- PRTGROUP or SPRTGROUP
- USERGROUP

Tip: GROUPDETAIL can greatly increase the number of SMF records generated for each Telnet profile. Use only if each individual member of a group is needed in the records.

SNAEXT statement

Use the SNAEXT parameter statement to enable negotiation for contention resolution and SNA sense functions for TN3270E connections. If NOSNAEXT is specified, Telnet does not negotiate these SNA functional extensions.

Telnet is initialized with a value of SNAEXT.

The SNAEXT and NOSNAEXT statements can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

The NOSNAEXT statement is useful in the unlikely case there are a significant number of clients that cannot tolerate the negotiation of these functions. Most clients do not have a problem with the SNAEXT specification in Telnet, but, in the unlikely case that some do, specify and map NOSNAEXT to that set of clients.

Syntax

```
►►──┬──────────┬──────────────────────────────────────────────────►◄
    ├─SNAEXT───┤
    └─NOSNAEXT─┘
```

Parameters

This statement has no parameters.

SSLTIMEOUT statement

Use the SSLTIMEOUT parameter statement to provide a unique timeout value for SSL handshake processing. This timeout limits the time SSL handshake processing waits for a client response.

Telnet is initialized with a SSLTIMEOUT value of 5.

The SSLTIMEOUT statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─SSLTIMEOUT 5───────┬──────────────────────────────────────────────►◄
    └─SSLTIMEOUT ──sec──┘
```

Parameters

5 Telnet is initialized with an SSLTIMEOUT value of 5 seconds.

sec
Sets the SSLTIMEOUT time to a specified number of seconds in the range 1 - 864 00.

SSLV2 and NOSSLV2 statements

Use the SSLV2 parameter statement to enable the SSLV2 protocol to be used on SECUREPORT connections. If NOSSLV2 is specified, the SECUREPORT connection supports SSLV3 or Transport Layer Security (TLS) only.

Telnet is initialized with a value of NOSSLV2.

The SSLV2 and NOSSLV2 statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─NOSSLV2─┬─────────────────────────────────────────────────────────►◄
    └─SSLV2 ──┘
```

Parameters

This statement has no parameters.

SSLV3 and NOSSLV3 statements

Use the SSLV3 parameter statement to enable the SSLV3 protocol to be used on SECUREPORT connections. If NOSSLV3 is specified, the SECUREPORT connection supports SSLV2 or Transport Layer Security (TLS) only.

Telnet is initialized with a value of NOSSLV3.

The SSLV3 and NOSSLV3 statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬──────────┬──────────────────────────────────────────────────────►◄
    ├─NOSSLV3─┤
    └─SSLV3───┘
```

Parameters

This statement has no parameters.

TCPIPJOBNAME statement

Use the TCPIPJOBNAME parameter statement to give the Telnet port affinity to the specified TCPIP stack. If NOTCPIPJOBNAME is specified, the port is an undirected port and binds with all stacks that have the port available.

Telnet is initialized with a value of NOTCPIPJOBNAME.

Restriction: You cannot change the TCPIPJOBNAME statement in a subsequent Telnet OBEYFILE command; you must restart Telnet to change its stack affinity.

The TCPIPJOBNAME and NOTCPIPJOBNAME statements can be coded in the TELNETGLOBALS statement block only. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬─────────────────────────────────┬───────────────────────────────►◄
    ├─TCPIPJOBNAME ──tcp_stack_name─┤
    └─NOTCPIPJOBNAME─────────────────┘
```

Parameters

tcp_stack_name
: The name of the TCPIP stack to which the Telnet port binds.

TELNETDEVICE statement

Use the TELNETDEVICE parameter statement to specify a logmode for a device type. This statement accepts two logmodes:

- TN3270 connections
- TN3270E connections

Syntax

```
►►──┬──────────────────────────────────────────────────────────────────────┬──────►◄
    └─TELNETDEVICE──telnet_device_type──┬─tn3270_logmode,tn3270e_logmode─┬─┘
                                        ├─tn3270_logmode─────────────────┤
                                        └─,tn3270e_logmode───────────────┘
```

Parameters

telnet_device_type
The type of Telnet device. See Table 35 for accepted device types.

tn3270_logmode
The logmode name used on TN3270 connections for the specified telnet_device_type.

tn3270e_logmode
The logmode name used on TN3270E connections for the specified telnet_device_type.

Device type and logmode table

Table 35. Device type and logmode table

Telnet device type	TN3270 logmode entry	TN3270E logmode entry
IBM-3277	D4B32782	Not applicable
IBM-3278-2-E	NSX32702	SNX32702
IBM-3278-2	D4B32782	SNX32702
IBM-3278-3-E	NSX32702	SNX32703
IBM-3278-3	D4B32783	SNX32703
IBM-3278-4-E	NSX32702	SNX32704
IBM-3278-4	D4B32784	SNX32704
IBM-3278-5-E	NSX32702	SNX32705
IBM-3278-5	D4B32785	SNX32705
IBM-3279-2-E	NSX32702	SNX32702
IBM-3279-2	D4B32782	SNX32702
IBM-3279-3-E	NSX32702	SNX32703
IBM-3279-3	D4B32783	SNX32703
IBM-3279-4-E	NSX32702	SNX32704
IBM-3279-4	D4B32784	SNX32704
IBM-3279-5-E	NSX32702	SNX32705
IBM-3279-5	D4B32785	SNX32705
IBM-3287-1	Not applicable	D6328904
IBM-DYNAMIC	D4C32XX3	D4C32XX3
LINEMODE	INTERACT	Not applicable
TRANSFORM	D4B32782	Not applicable

Telnet is initialized to the logmode names listed in Table 35. All the named logmode entries are defined to VTAM in the default logmode table, ISTINCLM. The TN3270 logmodes are non-SNA, and the TN3270E logmodes are SNA. For more details, see z/OS Communications Server: SNA Resource Definition Reference.

The TELNETDEVICE parameter statement can be coded in TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement blocks. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Telnet supports non-SNA (LU0) and SNA (LU2) terminal sessions. Telnet supports SNA character stream (SCS) (LU1) and 3270 data character stream (DCS) (LU3) printer sessions.

The specified logmode name can be an IBM-supplied logmode or user-created. If user-created, the BIND characteristics must be compatible with the LU type. TN3270 and TN3270E connections support either non-SNA or SNA BINDs.

The LOGMODE name NONE prevents Telnet from specifying a LOGMODE request with the REQSESS.

Telnet cannot verify that the logmode specified is valid at configuration time. Problems are detected at run time. For more information about logmodes, see z/OS Communications Server: SNA Resource Definition Reference.

TESTMODE statement

An operator can use the TESTMODE statement to test the new statements for a port without applying them. All the processing and checking is done for an actual port update, but at the end of the process, instead of applying the new statements, all data structures for that port are released. TESTMODE applies only to the port that is defined in the TELNETPARMS section where it is coded and not to the entire profile. If this statement is not coded, the profile for the port becomes the current profile when it is processed.

TESTMODE can be coded only in the TELNETPARMS statement block.

With the TESTMODE statement coded in all of the TELNETPARMS blocks, a Telnet administrator can issue a VARY TCPIP,,OBEYFILE command for a profile data set and can determine whether there are any syntax or semantic errors without concern for applying a profile that is not valid. TESTMODE profiles can be processed as often as necessary.

The TESTMODE statement can be specified in the initial startup profile. However, the end result is that the port is not opened and clients cannot connect. It would be as if no profile statements existed for that port.

Syntax

```
►►──┬──────────┬──────────────────────────────────────────────►◄
    └─TESTMODE─┘
```

Parameters

This statement has no parameters.

TIMEMARK statement

For a description of the TELNETPARMS TIMEMARK statement, see "SCANINTERVAL and TIMEMARK statements" on page 617.

TKOGENLU, TKOGENLURECON, and NOTKO statements

Use the TKOGENLU and TKOGENLURECON statements to enable an existing Telnet connection and its emulator, the target, to be taken over by a new Telnet connection and its emulator, the taker, under certain circumstances. NOTKOGENLU blocks generic takeover attempts and NOTKO blocks any takeover attempt of the target connection.

Two types of takeover exist:
- Specific LU takeover
- Generic LU takeover

The way a target can be taken over is defined in the profile associated with the target connection. The target can be set up to allow either or both takeover methods. The taker determines which takeover method is tried. If the taker specifies an LU name, a specific LU takeover is attempted, and the target must allow either TKOSPECLU or TKOSPECLURECON. If the taker specifies no LU name, a generic LU takeover is being attempted, and the target must allow either TKOGENLU or TKOGENLURECON.

When the profile indicates that generic takeover is allowed, Telnet saves the LU name of the first connection for each unique client identifier for all connections that allow generic takeover. Use generic takeover when there is only one connection per client identifier.

When generic takeover is allowed and a new generic connection request arrives, Telnet checks to determine whether the new connection client identifier already is already associated with an LU name. If client identifier does, Telnet attempts to take over the connection associated with that LU name. Telnet LU lookup suspends the new connection request. After the new connection is suspended, a TIMEMARK is sent to the original connection that is using the requested LU name. After the specified period of time, Telnet checks whether or not there was a response to the TIMEMARK. If a response or any data is received by the original connection since the TIMEMARK was sent out, Telnet fails the new connection takeover attempt and then assigns the next available LU name to the new connection. If no response is received, the target connection is dropped and the new taker connection is established with the saved LU name. If TKOGENLU is in effect, the session is also dropped. If TKOGENLURECON is in effect, the session is transferred to the taker connection.

Restriction: If the NOTKOGENLU or NOTKO statement is specified, generic takeover of the target cannot be performed.

Telnet is initialized with a value of NOTKO.

The TKOGENLU, TKOGENLURECON, NOTKOGENLU and NOTKO statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Generic LU takeover and specific LU takeover can coexist. TKOGENLU and TKOGENLURECON are mutually exclusive. TKOSPECLU and TKOSPECLURECON are mutually exclusive.

When the TKOGENLU or TKOGENLURECON statement is specified and one connection exists for a client, any additional connection request first tries takeover using the first connection LU name. After the takeover request fails, Telnet continues generic LU lookup. Therefore, all additional connection requests are delayed by the takeover time specified.

In some cases, the TKOGENLURECON session cannot be maintained. See z/OS Communications Server: IP Configuration Guide, Advanced Application topics for details.

Syntax

```
►►──┬─────────────────────────────────────────────────────────────────────────────────┬──────────►◄
    ├─NOTKOGENLU──────────────────────────────────────────────────────────────────────┤
    ├─ NOTKO──────────────────────────────────────────────────────────────────────────┤
    ├─TKOGENLU──sec───────────────────────────────────────────────────────────────────┤
    │                      ┌─NOKEEPONTMRESET─┐  ┌─NOSAMEIPADDR─┐  ┌─NOSAMECONNTYPE─┐  │
    └─TKOGENLURECON──sec───┼─────────────────┼──┼──────────────┼──┼────────────────┼──┘
                           └─KEEPONTMRESET───┘  └─SAMEIPADDR───┘  └─SAMECONNTYPE───┘
```

Parameters

sec
Number of seconds Telnet waits before checking to determine whether a response was received from the original client. The range is 0 - 99 999 999. Zero is a special case value. If you code 0 in the *sec* field, Telnet always performs the takeover, whether the original session is active or not.

KEEPONTMRESET
If a reset is received from the target during takeover, the session is saved and transferred to the taker. If the KEEPONTMRESET parameter is not specified, or if the NOKEEPONTMRESET parameter is specified, the session is dropped if a reset is received from the target.

SAMEIPADDR
Ensures that the taker has the same IP address as the target. If the SAMEIPADDR parameter is not specified, or if the NOSAMEIPADDR parameter is specified, a taker with a different IP address can take over the target. The changed IP address is not forwarded to the application, which could cause possible reporting errors.

SAMECONNTYPE
Ensures that the taker has the same basic or secure connection type as the target. If the SAMECONNTYPE parameter is not specified, or if the NOSAMECONNTYPE parameter is specified, a taker with a secure connection can take over a target with a basic connection. The original connection type is forwarded to the application as part of the CINIT CV64 information. The changed connection type from basic to secure is not forwarded to the application, which could cause possible reporting errors.

TKOSPECLU, TKOSPECLURECON, and NOTKO statements

Use the TKOSPECLU and TKOSPECLURECON statements to enable an existing Telnet connection and its emulator, the target, to be taken over by a new Telnet connection and its emulator, the taker, under certain circumstances. NOTKOSPECLU blocks specific takeover attempts and NOTKO blocks any takeover attempt of the target connection.

The following types of takeover exist:

- Specific LU takeover
- Generic LU takeover

The way a target can be taken over is defined in the profile associated with the target connection. The target can be set up to allow either or both takeover methods. The taker determines which takeover method is tried. If the taker specifies an LU name, a specific LU takeover is attempted, and the target must allow either TKOSPECLU or TKOSPECLURECON. If the taker does not specify an LU name, a generic LU takeover is being attempted, and the target must allow either TKOGENLU or TKOGENLURECON.

When specific LU takeover is allowed, Telnet LU lookup suspends a new connection request that specifies an already active LU name. After the new connection is suspended, a TIMEMARK is sent to the original connection that is using the requested LU name. After the specified period of time, Telnet checks whether there was a response to the TIMEMARK. If a response or any data is received by the original connection since the TIMEMARK was sent out, Telnet fails the new connection takeover attempt by indicating the LU name is already in use. If no response is received, the target connection is dropped and the new taker connection is established with the specified LU name. If TKOSPECLU is in effect, the session is also dropped. If TKOSPECLURECON is in effect, the session is transferred to the taker connection.

Restriction: If the NOTKO statement or the NOTKOSPECLU statement is specified, specific takeover of the target cannot be performed.

Telnet is initialized with a value of NOTKO.

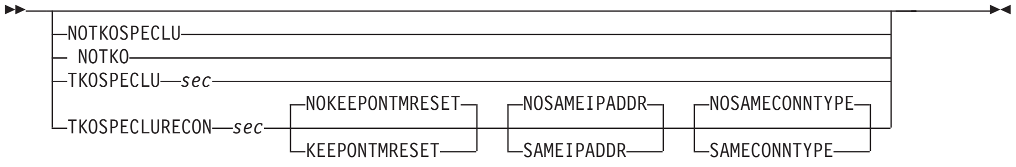
The TKOSPECLU, TKOSPECLURECON, NOTKOSPECLU and NOTKO statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See "Rules for Telnet parameter statements and security parameters" on page 593 for more information about the hierarchy of parameter values.

Generic LU takeover and specific LU takeover can coexist. The TKOGENLU and TKOGENLURECON statements are mutually exclusive. The TKOSPECLU and TKOSPECLURECON statements are mutually exclusive.

Requirements:
- To take over the session using the TKOSPECLU or TKOSPECLURECON statement, the new connection must specify the LU name. If administrators want to use this function for a more general purpose, code the @@LUNAME character substitution in the MSG10 screen so end users know their LU name if they need to issue a takeover. Also, some clients display the LU name assigned by Telnet.
- You must have a specific LU pool for a specific LU connection request. If you are switching from generic LU connection requests to specific LU connection requests and are using the DEFAULTLUS pool, a DEFAULTLUSSPEC pool must be defined.

In some cases, the TKOSPECLURECON session cannot be maintained. See z/OS Communications Server: IP Configuration Guide, Advanced Application topics for details

Syntax



Parameters

sec
: Number of seconds Telnet waits before checking whether a response was received from the original client. Valid values are in the range is 0 - 99 999 999. The value 0 is a special case value. If you code 0 in the *sec* field, Telnet always performs the takeover, whether the original session is active or not.

KEEPONTMRESET
: If a reset is received from the target during takeover, the session is saved and transferred to the taker. Without KEEPONTMRESET or if NOKEEPONTMRESET is specified, the session is dropped if a reset is received from the target.

SAMEIPADDR
: Ensures that the taker has the same IP address as the target. Without SAMEIPADDR or if NOSAMEIPADDR is specified, a taker with a different IP address can take over the target. The changed IP address is not be forwarded to the application, which could cause possible reporting errors.

SAMECONNTYPE
: Ensures that the taker has the same basic or secure connection type as the target. If the SAMECONNTYPE parameter is not specified, or if the NOSAMECONNTYPE parameter is specified, a taker with a secure connection can take over a target with a basic connection. The original connection type is forwarded to the application as part of the CINIT CV64 information. The changed connection type from basic to secure is not forwarded to the application, which could cause possible reporting errors.

TN3270E statement

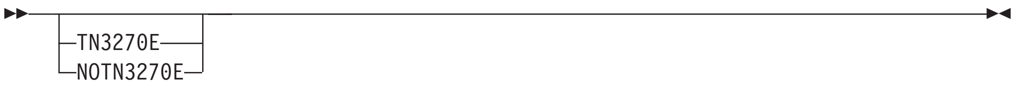
Use the TN3270E parameter statement to allow TN3270E functions to be negotiated by Telnet. If NOTN3270E is specified, all TN3270E functions, such as printer support and client response, are disabled.

Telnet is initialized with a value of TN3270E.

The TN3270E and NOTN3270E statements can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

The NOTN3270E value is useful in the unlikely case there are a significant number of clients that cannot tolerate negotiating for a TN3270E connection. Most clients do not have a problem with the TN3270E specification in the server, but, in the unlikely case that some do, specify and map NOTN3270E to that set of clients.

Syntax



Parameters

This statement has no parameters.

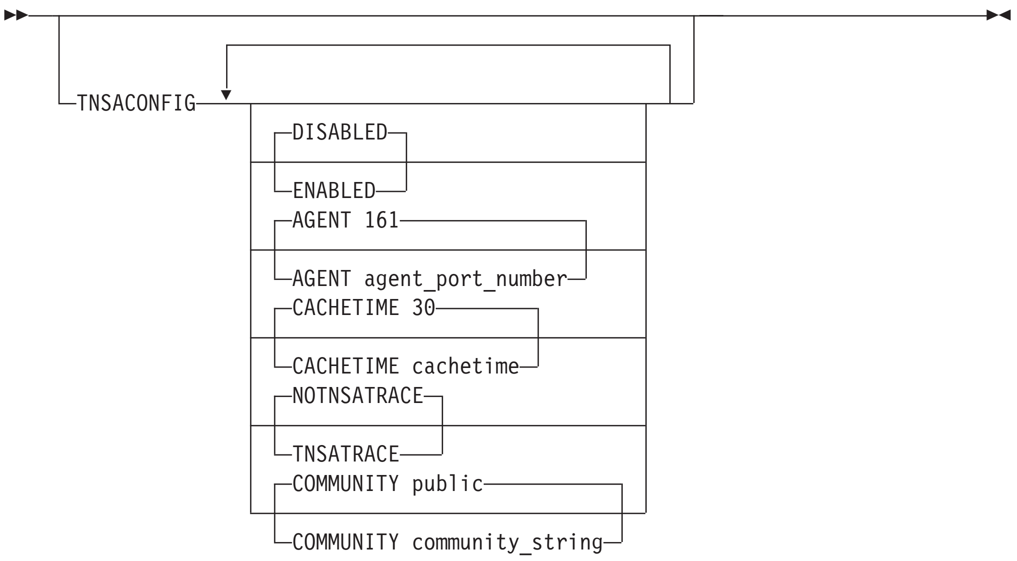
TNSACONFIG statement

Use the optional TNSACONFIG statement to configure the SNMP TN3270E Telnet subagent.

The Telnet defaults are:
- DISABLED
- AGENT 161
- COMMUNITY PUBLIC
- NOTNSATRACE
- CACHETIME 30

Restrictions:
- The TNSACONFIG statement can be coded only in the TELNETGLOBALS statement block.
- No parameters can change while the Telnet subagent is active. To make a change, the Telnet subagent must be disabled and then enabled again with the new parameter values.

Syntax



Parameters

AGENT
A port number in the range 1 - 65 535 used in establishing communication with the SNMP agent. For the Telnet SNMP subagent to communicate with the z/OS CS SNMP agent, the port number specified must match the port number specified on the -p parameter when the SNMP agent is started. See “OSNMPD parameters” on page 1243 for a description of how to specify the port when the SNMP agent is started.

CACHETIME
Amount of time in seconds to elapse before rebuilding the MIB object tables. The valid range is 0 - 99 999 999.

COMMUNITY
A character string 1- 32 characters in length used as the community name (or password) in establishing contact with the SNMP agent. Because the community name is case sensitive, it is not converted to uppercase by profile processing. It cannot contain any imbedded white space or control characters (such as blank, tab, end of line, or end of file) and cannot contain any imbedded semicolons (semicolons are treated as comment delimiters). For the Telnet SNMP subagent to communicate with the z/OS Communications Server SNMP agent, the community name specified on the COMMUNITY keyword must match one that is defined in the PW.SRC or SNMPD.CONF data set used by the SNMP agent or specified on the -c parameter when the SNMP agent is started.

For more information about how the community name is used to permit access to the SNMP agent, see Step 1: Configure the SNMP agent (OSNMPD), in z/OS Communications Server: IP Configuration Guide.

DISABLED|ENABLED
DISABLED specifies that you do not require any of the SNMP MIB data supported by the Telnet subagent. By default, the Telnet SNMP subagent is not started during Telnet initialization. If specified using the VARY TCPIP,,OBEYFILE command, this statement indicates that the currently active Telnet subagent task should be terminated. SNMP MIB objects supported by the z/OS CS SNMP agent and subagents other than the Telnet SNMP subagent are still available. For information about which MIB objects are supported by the SNMP agent and subagent, see z/OS Communications Server: IP User's Guide and Commands.

ENABLED indicates that the Telnet SNMP subagent should be started at the completion of profile processing, either of the initial profile or of the data set referenced on a VARY TCPIP,,OBEYFILE command.

NOTNSATRACE|TNSATRACE
TNSATRACE generates trace points throughout Telnet subagent processing in addition to tracing data passed between the Telnet subagent and the agent, Telnet, and TCP/IP stack. The trace data is written to the syslog daemon.

UNLOCKKEYBOARD statement

Use the UNLOCKKEYBOARD statement to customize an unlock keyboard sequence being forwarded to the client from the host application.

Telnet is initialized with a value of UNLOCKKEYBOARD BEFOREREAD TN3270BIND.

The UNLOCKKEYBOARD statement can be coded in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP statement block. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►──┬──────────────────────────────────────────────────────────┬──────────────────────►◄
    │                 ┌─BEFOREREAD─┐   ┌─TN3270BIND───┐         │
    └─UNLOCKKEYBOARD──┼────────────┼───┼──────────────┼─────────┘
                      ├─BEFOREREAD─┤   ├─TN3270BIND───┤
                      └─AFTERREAD──┘   └─NOTN3270BIND─┘
```

Parameters

BEFOREREAD
: Indicates that when conditions warrant, an unlock keyboard sequence is sent to the client before forwarding a read command from the host application.

AFTERREAD
: Indicates that when conditions warrant, an unlock keyboard sequence is sent to the client after forwarding a read command from the host application.

TN3270BIND
: Indicates that when a BIND from the VTAM host application is received by Telnet for a TN3270 connection, a clear screen and an unlock keyboard is sent to the client.

NOTN3270BIND
: Indicates that when a BIND from the VTAM host application is received by Telnet for a TN3270 connection, neither a clear screen, nor an unlock keyboard are sent to the client.

XCFGROUP statement

Use the optional XCFGROUP statement block to define and join the Telnet XCF group and to provide parameter values for the shared LU name management services. When an XCF Telnet joins the Telnet XCF group, that XCF Telnet can support the LU name server (LUNS) function, the LU name requester (LUNR) function, or both, based on which parameters are coded in the XCFGROUP statement and the BEGINVTAM block.

If the XCFGROUP statement is not coded, Telnet is initialized with the NOJOIN parameter.

The XCFGROUP statement can be coded only in the TELNETGLOBALS statement block. See “Rules for Telnet parameter statements and security parameters” on page 593 for more information about the hierarchy of parameter values.

Syntax

```
►►─┬──────────────────────────────────────────────────────────────────┬──►◄
   │            ┌──────────────────────────────────────┐              │
   │            ▼                                      │              │
   └─XCFGROUP──┬┴──────────────────────────────────────┴──ENDXCFGROUP─┘
               │ ┌─JOIN──────────┐                     │
               ├─┴───────────────┴─────────────────────┤
               │ └─┬─NOJOIN──┬───┘                     │
               │   └─JOIN────┘                         │
               ├─SUBPLEX suffix────────────────────────┤
               │ ┌─XCFMONITOR 60───┐                   │
               ├─┴─────────────────┴───────────────────┤
               │ └─XCFMONITOR sec──┘                   │
               │ ┌─CONNECTTIMEOUT 60───┐               │
               ├─┴─────────────────────┴───────────────┤
               │ └─CONNECTTIMEOUT sec──┘               │
               │ ┌─RECOVERYTIMEOUT 60───┐              │
               ├─┴──────────────────────┴──────────────┤
               │ └─RECOVERYTIMEOUT sec──┘              │
               │                  ┌─PRIMARY ─RANK 255──┐
               └─LUNS ─ipaddr port─┼───────────────────┼─┘
                                  │          ┌─RANK 255─┐   │
                                  ├─PRIMARY──┴──────────┴───┤
                                  │          └─RANK num─┘   │
                                  │         ┌─RANK 1───┐    │
                                  └─BACKUP──┴──────────┴────┘
                                            └─RANK num─┘
```

Parameters

JOIN
Specifies that this Telnet should join the Telnet XCF group. Telnet uses only the XCF features available at the XCF-local level of functionality; the system does not need to be a member of a sysplex. Members of the XCF group are visible from any member of the group in XCF group status displays. A Telnet member of the XCF group is a potential LUNS if the LUNS parameter is coded. A member is a LUNR if shared LU names are defined in the BEGINVTAM block.

NOJOIN
Specifies that this Telnet should not join the Telnet XCF group. XCF group status displays are not available from this Telnet and this Telnet is not be visible in XCF group status displays from any member of the XCF group. This Telnet cannot become a LUNR and cannot define shared LU name objects.

SUBPLEX *suffix*
Specifies the character suffix (1 - 4 characters in length) to use for the Telnet XCF group name and ENQUE names to partition a sysplex into multiple Telnet subplexes. Telnet is initialized to use the string EZZTLUNS. The specified suffix is right-aligned and overlays the end of this string to form unique subplex strings. For example, if the suffix value is 23, Telnet joins XCF group EZZTLU23.

XCFMONITOR *sec*
Sets the XCF monitor interval to the number of seconds that a LUNR attempts to establish a connection to the LUNS before quiescing its LUNR capabilities. At the specified time interval, Telnet checks the health of the LUNS, LUNR, and XCF Telnet tasks and checks the health of the connection between the LUNS and LUNR. If any of these tasks or connections appear to be unresponsive, message EZZ6099I is issued and the X indicator is set to on under the PDMON column in the XCFGROUP display.

The valid values for this timer are in the range 10 - 3600.

CONNECTTIMEOUT *sec*
This parameter applies only to LUNR. Sets the monitor interval to the number

seconds that a LUNR attempts to establish a connection to the LUNS before quiescing its LUNR capabilities. If the LUNR has not been able to connect to the LUNS within the amount of time, then the LUNR has not been able to connect with the LUNS, the LUNR drops all connections that are waiting in negotiation for an LU name and quiesces all ports that have shared groups. This action frees clients to reconnect to a working LUNR. The value 0 disables the connect timeout interval. Valid values are 0 or an integer in the range 10 - 99 999 999.

RECOVERYTIMEOUT *sec*
This paramete applies only to LUNRr. Sets the number of seconds that a LUNR attempts to establish a connection to the recovering LUNS before dropping connections using shared LU names. When a LUNS takeover occurs and a new LUNS becomes available, each Telnet LUNR repeatedly attempts to connect to the new LUNS. The recovery of the LUNS cannot complete until all LUNRs have recognized the new LUNS and all LUNRs that have allocated LU names have connected to the new LUNS and re-registered all previously allocated shared LU names. If the LUNR does not successfully connect within the specified time, it drops all existing client connections that are using shared LU names. The recovery of the LUNS completes and shared shared LU name management resumes without this LUNR. The value 0 disables the recovery timeout interval.

Valid values are 0 or an integer in the range 10 - 99 999 999.

LUNS PRIMARY
Specifies that this Telnet becomes the active LUNS at job initiation if there is not already an active LUNS. If there is already an active LUNS, this Telnet becomes a standby. Telnet must join the XCF group to be a LUNS.

LUNS BACKUP
Specifies that this Telnet becomes a standby LUNS at job initiation. Telnet must join the XCF group to be a LUNS.

ipaddr
Specifies the IP address that this Telnet listens on for shared LU name management requests when the address becomes the active LUNS.

port
Specifies the port that this Telnet listens on for shared LU name management requests when the port becomes the active LUNS.

RANK
Specifies the takeover rank of this LUNS when it is in standby mode and the active LUNS fails. The standby LUNS with the highest rank becomes the new LUNS. If there is more than one standby LUNSs with the same rank, they compete for a sysplex scope ENQUEUE. The winner becomes the new LUNS, and the others return to standby mode. Valid values are in the range 1 - 255.

Telnet mapping statements in the Telnet profile

Mapping statements for Telnet are specified in the BEGINVTAM block. All mapping statements are optional for the BEGINVTAM block.

Some statements combine mapping and object functions. For example, DEFAULTLUS defines the LU GROUP Object and implicitly maps the group to the NULL Client Identifier. ALLOWAPPL defines the security level of application Objects and optionally provides LU mapping function.

Table 36 provides a list of Telnet mapping statements and the location of more information.

Table 36. Telnet mapping statements

Statement	Mapping statement	Client identifier	Object	See page
ALLOWAPPL	X		X	"ALLOWAPPL statement" on page 640
DEFAULTAPPL	X			"DEFAULTAPPL statement" on page 641
DEFAULTLUS and SDEFAULTLUS	X		X	"DEFAULTLUS or SDEFAULTLUS statement" on page 642
DEFAULTLUSSPEC and SDEFAULTLUSSPEC	X		X	"DEFAULTLUSSPEC or SDEFAULTLUSSPEC statement" on page 643
DEFAULTPRT and SDEFAULTPRT	X		X	"DEFAULTPRT or SDEFAULTPRT statement" on page 644
DEFAULTPRTSPEC and SDEFAULTPRTSPEC	X		X	"DEFAULTPRTSPEC or SDEFAULTPRTSPEC statement" on page 645
DESTIPGROUP		X		"DESTIPGROUP statement" on page 645
HNGROUP		X		"HNGROUP statement" on page 646
INTERPTCP	X			"INTERPTCP statement" on page 647
IPGROUP		X		"IPGROUP statement" on page 648
LINEMODEAPPL	X			"LINEMODEAPPL statement" on page 649
LINKGROUP		X		"LINKGROUP statement" on page 650
LUGROUP and SLUGROUP			X	"LUGROUP or SLUGROUP statement" on page 651
LUMAP	X			"LUMAP statement" on page 652
MONITORGROUP			X	"MONITORGROUP statement" on page 654
MONITORMAP	X			"MONITORMAP statement" on page 655
PARMSGROUP			X	"PARMSGROUP statement" on page 656
PARMSMAP	X			"PARMSMAP statement" on page 656
PORT				"PORT statement" on page 657
PRTDEFAULTAPPL	X			"PRTDEFAULTAPPL statement" on page 657
PRTGROUP and SPRTGROUP			X	"PRTGROUP or SPRTGROUP statement" on page 658
PRTMAP	X			"PRTMAP statement" on page 660
RESTRICTAPPL	X		X	"RESTRICTAPPL statement" on page 661
USERGROUP		X		"USERGROUP statement" on page 663
USSTCP	X			"USSTCP statement" on page 664

Rules: Observe the following rules for BEGINVTAM statements:

- If the BEGINVTAM block represents more than one port, the first statement in the BEGINVTAM block must be the port designation statement.

- Telnet must have an application and Telnet LUs defined in order to connect to a host application.
- Object and Client Identifier group names can include any printable character except those in Table 37:

Table 37. Object and Client Identifier group name printable character exceptions

Character	EBCDIC	Description
.	4B	Period
*	5C	Asterisk
;	5E	Semicolon
,	6B	Comma
=	7E	Equal

- Any Object or Client Identifier group name must be defined before it can be specified on a mapping statement. Otherwise, the group name is interpreted as a linkname.
- LUGROUPs and PRTGROUPs must be mapped to a Client Identifier to be used.
- If one element in a group is not valid, Telnet flags the element that is not valid and processes the statement as if the element were not part of the statement. If all elements are not valid, Telnet issues a debug message indicating the GROUP is empty.
- The second instance of the Client Identifier in the second group is ignored and a message is issued. For example:

  ```
  IPGROUP ABC 1.1.1.1 2.2.2.2 ENDIPGROUP
  IPGROUP XYZ 2.2.2.2 3.3.3.3 ENDIPGROUP
  ```

  The second IPGROUP statement generates a debug warning message indicating that “2.2.2.2” is already defined in an IPGROUP.
- An IPGROUP with a subnet mask of 0.0.0.0:0.0.0.0 specified matches all clients.

Rules for LU name specification

Rules: Observe the following rules for LU name specification:

- The first character must be in the range A through Z, @, #, or $. In addition, remaining characters can also be numeric (any single digit 0 through 9). Unprintable characters are not allowed. If a name that is not valid is found, an error message is issued and the statement is ignored.
- LUs can be defined as a range. Use the following syntax to specify a range of LUs:

  ```
  ►►──LowerRange..UpperRange──┬─────────────┬──────────────────────────►◄
                              └─..rangerule─┘
  ```

  – No spaces are allowed within a range definition.
  – *UpperRange* must be greater than the *LowerRange*.
  – The lengths of *LowerRange* or *UpperRange*, and *rangerule* must be the same and each must be less than or equal to eight characters.
  – All LUs in the range must be valid and defined to VTAM for a successful session.
  – The number of LU names in one range is limited to 4 294 967 295. The total number of LU names in the group is also limited to 4 294 967 295. Storage is not used until the LU name is assigned to the connection.

– The *rangerule* represents the variant used for wildcarding. For example:

```
TCP000A0..TCP9F$ZZ..FFFNX?AB
```

where:

F The position is fixed and does not change.

A Alphabetic range.

N Numeric range.

B Alphanumeric range.

X Hexadecimal range.

? Alphanumeric including national characters @, #, and $.

If an incorrect range definition is parsed, it is ignored and a debug warning message is issued.

Result: The range specification AB100..CB299..AFNNN defines AB100-AB999 (900), BB000-BB999 (1000) and CB000-CB299 (300) (2200 names). If a specification of AB100-AB299, BB100-BB299 and CB100-CB299 (600 names) is desired, then two range specifications are required: AB100..CB199..AFFNN AB200..CB299..AFFNN.

See z/OS Communications Server: IP Configuration Guide for LU range usage examples.

- If the range rule is omitted, Telnet assumes the following style, where the LowerRange and UpperRange values must be all numeric or all alphabetic:

```
LuBase+LowerRange..LuBase+UpperRange
```

Client identifier types and definitions

Table 38 shows the Client Identifier types and their definitions available for use on mapping statements.

Table 38. Client identifier types and definitions

Client identifier type	Definition
USERID	The client User ID derived from the client certificate at connection time when ClientAuth SAFcert is specified on an SSL connection.
HOSTNAME	The completely qualified client host name.
IPADDR	The client IP address expressed in dotted decimal form. This can be an IPv4 address only.
USERGRP	The USERGROUP name that contains exact or wildcard client user IDs.
HNGRP	The HNGROUP name that contains exact or wildcard client host names.
IPGRP	The IPGROUP name that contains exact or subnetted client IP addresses.
DESTIP	The destination IP address expressed in dotted decimal form.
LINKNAME	The link or interface name defined by the LINK or INTERFACE statement in PROFILE.TCPIP.
DESTIPGRP	The DESTIPGROUP name that contains exact or subnetted destination IP addresses.
LINKGRP	The LINKGROUP object name that contains exact or wildcard link or interface names.

Table 38. Client identifier types and definitions (continued)

Client identifier type	Definition
NULL	Not coded, but listed here for completeness. This Client Identifier type indicates that no Client Identifier was specified. This is valid for the DEFAULTAPPL, LINEMODEAPPL, USSTCP, and INTERPTCP mapping statements. It is the implied Client Identifier for the DEFAULTLUS, DEFAULTLUSSPEC, DEFAULTPRT, and DEFAULTPRTSPEC Object statements.

See z/OS Communications Server: IP Configuration Guide for Object selection priority based on Client Identifiers.

Rules for client identifier specification

Observe the following rules for client identifier specification:

- When the Client Identifier is a single entity on a mapping statement rather than part of a group, no wildcarding is allowed.
- A *group name* must be defined with the appropriate statement before it can be specified on a MAPPING statement. Otherwise, the name is assumed to be a link or interface name.
- User ID and destination IP address require the *clid_type* keyword to correctly identify the Client Identifier. If *clid_type* is not used, a user ID Client Identifier is assumed to be a link or interface name and a destination IP address Client Identifier is assumed to be the traditional client (source) IP address.
- Client Identifiers of a particular type, such as IP address or host name, can be defined within only one group of that type. If the Client Identifier is defined in more than one group, a debug warning message is issued showing the Client Identifier that is ignored and the name of the owning group. No error is issued if a Client Identifier is listed twice in the same group.

  See z/OS Communications Server: IP Configuration Guide for exact mapping rules.

Rules for host name specification

Observe the following rules for host name specification:

- Host name specification requires that Telnet be able to resolve a host name from an IP address by use of the resolver. To do this, a valid TCPIP.DATA data set must be provided. For overview information about TCP/IP application configuration files, see z/OS Communications Server: IP Configuration Guide for a description of how TCPIP.DATA is located. Telnet uses the native MVS sockets search order to find a resolver. Neither the z/OS environmental variable (Resolver_Config) nor the /etc/resolv.conf z/OS UNIX is used when searching for TCPIP.DATA.
- The Telnet client IP address and port are automatically added to the z/OS Communications Server SNA displays. An HNGROUP statement is required if you also want the DNS name of the client. If you are mapping objects using host names, the DNS names of the Telnet clients is provided to the z/OS Communications Server SNA displays automatically. This occurs automatically because the names must have been resolved for mapping purposes. If you are not mapping by host names, but want to have Telnet client host names provided to the z/OS Communications Server SNA displays, add an HNGROUP name and ENDHGGROUP name to your Telnet profile. Choose an unused host name

(such as AA.AA). If you add the HNGROUP statement to get DNS name resolution, some delay might occur during connection processing for name resolution.

ALLOWAPPL statement

Use the *optional* ALLOWAPPL mapping and security statement to specify which VTAM application names clients can access and optionally, which LU names are valid.

Syntax

```
►►──ALLOWAPPL──application_name──┬────────────────┬──QSESSion──┬──────┬──┬─────────────────────────┬──────────►◄
                                 └─DISCONNECTABLE─┘            └─,sec─┘  │ ┌─────────────────────┐ │
                                                                         │ ▼                     │ │
                                                                         ├─LU── ──lu_name────────┴─┤
                                                                         └─LUG── ──lu_group_name───┘
```

Parameters

application_name
: The host application name, as specified in VTAMLST.

  Single-character position wildcards (%) are permitted anywhere in the application name and the multi-character wildcard (*) is permitted at the end of an application name. For example, A%CICS* allows connections to A1CICS01, A1CICS02, ABCICS4A, and so on. A single * allows all applications.

DISCONNECTABLE
: When DISCONNECTABLE is specified, VTAM notifies the application to disconnect, rather than log off a user, when the session is dropped.

QSESSion
: Indicates this application queues a session request when passing the session to another primary application. When Telnet receives an UNBIND of the new session, Telnet waits for a BIND to reestablish the original queued session.

sec
: When QSESSion is coded, this value determines the number of seconds Telnet waits before checking whether a BIND was received. The range is 1 - 99 999 999. If no BIND is received in the time specified, Telnet stops waiting and continues cleaning up the connection as if QSESSion had not been coded. There is no default value. If *sec* is not coded, the connection never checks whether a BIND is received. Telnet waits until a BIND is received or the connection is dropped.

LU *lu_name*
: The logical name of the Telnet terminal LU. This parameter allows you to optionally specify which terminal LUs can be used to establish a session with the named VTAM host application.

LUG *lu_group_name*
: The name of the LUGROUP or PRTGROUP. This option allows you to specify an LUGROUP or PRTGROUP, where any LU in the group can be used to establish a session with the named VTAM host application. If the same name defines both an LUGROUP and a PRTGROUP, the LUGROUP is used. The group can be a new group consisting of a combination of names or range list names from existing LUGROUPs and PRTGROUPs. This allows both terminals and printers to be on the same ALLOWAPPL statement.

Usage notes

- Applications that perform CLSDST PASS also require an ALLOWAPPL or RESTRICTAPPL statement for the target application.
- LU and LUG keywords are mutually exclusive. If both are specified in any order, only the last LUG is accepted and processed. If multiple LUG keywords are specified, only the last is accepted and processed.
- If the LU assigned to the connection is defined in LU groups mapped by both a LUG statement and an LUMAP/PRTMAP statement, neither LU group can be defined as an LU exit.

DEFAULTAPPL statement

Use the *optional* DEFAULTAPPL mapping statement to map the initial application to be tried when a Telnet client establishes a connection other than linemode. The application might be a particular VTAM application such as CICS® or could be a network solicitor or front-end menu system such as TPX. DEFAULTAPPL allows a user to establish a session with an application without having to know the actual VTAM name of the application.

Syntax

```
►►──DEFAULTAPPL──application_name──┬──────────────────────────────┬──────────────►
                                   ├─Client_Identifier────────────┤
                                   └─clid_type,Client_Identifier──┘

►──┬───────────┬──┬─────────┬──┬─────────┬──────────────────────────────────────►◄
   └─FIRSTONLY─┘  ├─LOGAPPL─┤  └─DEFONLY─┘
                  └─QINIT───┘
```

Parameters

`application_name`
: The host application name, as specified in VTAMLST. The *application_name* can be network qualified in the format of a 1- to 8-character name of the network ID separated by a period (.), followed by a 1- to 8-character application name.

`clid_type`
: Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See "Rules for client identifier specification" on page 639 for details.

`Client_Identifier`
: One of several Client Identifiers. See "Client identifier types and definitions" on page 638 for details. If no Client Identifier is specified, then it is considered the NULL Client Identifier.

FIRSTONLY
: When FIRSTONLY is specified, a solicitor or USSMSG10 screen is sent to the client after logoff from a default session when LUSESSIONPEND is coded. When FIRSTONLY is not specified, Telnet always requests a new session to the default application after logoff from the session when LUSESSIONPEND is coded. If LUSESSIONPEND is not coded, the connection is dropped.

LOGAPPL
: When LOGAPPL is specified, a session request to a host application that is not active is queued in VTAM instead of rejected. Telnet keeps the ACB open for the LU representing the client. When the application becomes active, VTAM initiates a session between the application and the Telnet LU.

QINIT
Indicates that session requests should be queued, and when logging off the default application, Telnet should redrive the default application instead of issuing a USSMSG10 or Solicitor screen.

DEFONLY
When DEFONLY is specified, the client is blocked from specifying any application name other than the one specified on the default application statement.

Usage notes

- Always map a unique Client Identifier on each DEFAULTAPPL statement. Otherwise, the last DEFAULTAPPL mapping for the Client Identifier is used.
- If a USS table is mapped to the Client based on a higher priority Client Identifier, the DEFAULTAPPL statement is ignored.

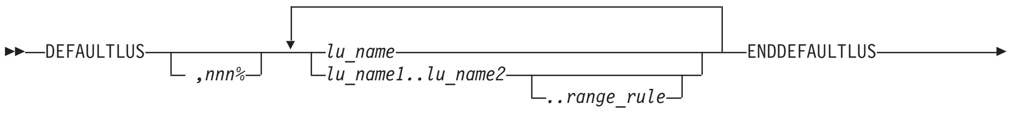
DEFAULTLUS or SDEFAULTLUS statement

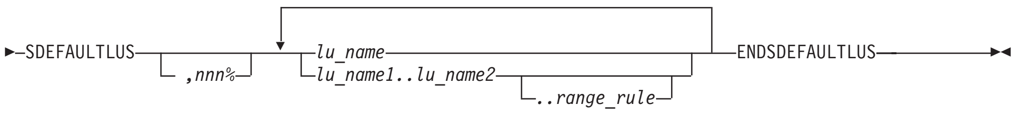
Use the *optional* DEFAULTLUS or SDEFAULTLUS object and mapping statement to define a list or range of LUs that have a default mapping to the NULL client identifier. This LU pool is used by a terminal emulator that is requesting a generic connection if no other LU group maps generically to the client. The SDEFAULTLUS statement defines shared LU names rather than private ones.

Restrictions:

- This Telnet must have joined the XCF group to define shared objects. An active Telnet LU name server (LUNS) must exist in order for the profile to be processed and for the shared LUs to be usable.
- A profile can have either DEFAULTLUS or SDEFAULTLUS defined, but not both.

Syntax





Parameters

nnn%
Checks the capacity remaining in the group when Telnet assigns an LU from that group. A message is issued when the specified percentage is reached. After the group exceeds the specified capacity, no other message is issued. After the capacity being used drops below 10 percent of the total capacity check amount, another capacity warning message is issued. Leave a blank space between DEFAULTLUS or SDEFAULTLUS and the comma (,) that is part of the capacity field.

lu_name
The name of the terminal LU.

lu_name1..lu_name2
A range of terminal LUs.

range_rule
The wildcard method used for each character position.

Usage notes

- See “Rules for LU name specification” on page 637 for LU name and LU range specification rules.

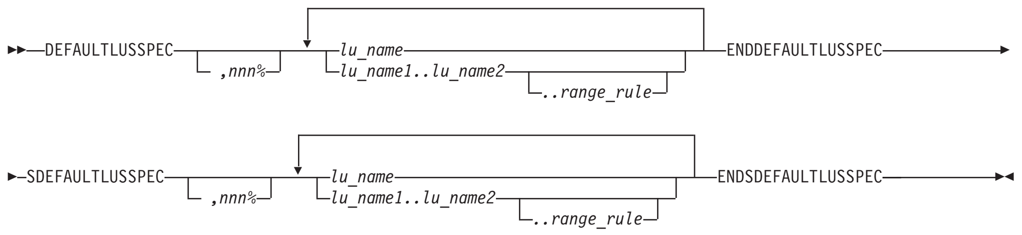
DEFAULTLUSSPEC or SDEFAULTLUSSPEC statement

Use the *optional* DEFAULTLUSSPEC or SDEFAULTLUSPEC object and mapping statements to define a list or range of LUs that have a default mapping to the NULL client identifier. This pool is used by a terminal emulator that is requesting a specific connection if no other LU group maps specifically or generically to the client. The SDEFAULTLUSSPEC statement defines shared LU names rather than private ones.

Restrictions:

- This Telnet must have joined the XCF group to define shared objects. An active Telnet LU name server (LUNS) must exist in order for the profile to be processed and for the shared LUs to be usable.
- A profile can have either DEFAULTLUSSPEC or SDEFAULTLUSSPEC defined, but not both.

Syntax



Parameters

nnn%
Checks the capacity remaining in the group when Telnet assigns an LU from that group. A message is issued when the specified percentage is reached. After the group exceeds the specified capacity, no other message is issued. After the capacity being used drops below 10 percent of the total capacity check amount, another capacity warning message is issued. Leave a blank space between DEFAULTLUS or SDEFAULTLUS and the comma (,) that is part of the capacity field.

lu_name
The name of the terminal LU.

lu_name1..lu_name2
A range of terminal LUs.

range_rule
The wildcard method used for each character position.

Usage notes

See “Rules for LU name specification” on page 637 for LU name and LU range specification rules.

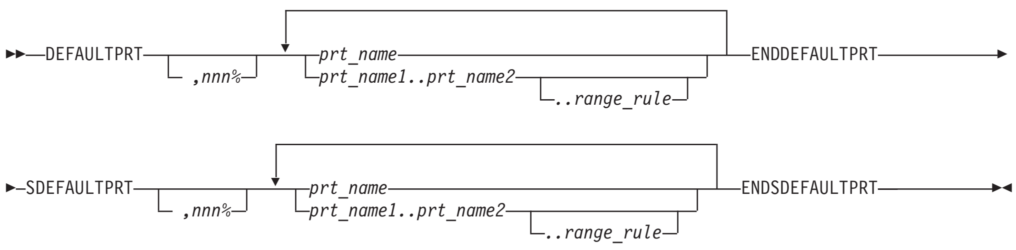
DEFAULTPRT or SDEFAULTPRT statement

Use the *optional* DEFAULTPRT or SDEFAULTPRT object and mapping statements to define a list or range of printer LUs that have a default mapping to the NULL client identifier. This LU pool is used by a printer emulator that is requesting a generic connection if no other printer LU group maps to the client. The SDEFAULTPRT statement defines shared LU names rather than private ones.

Restrictions:

- This Telnet must have joined the XCF group to define shared objects. An active Telnet LU name server (LUNS) must exist in order for the profile to be processed and for the shared LUs to be usable.
- A profile can have either DEFAULTPRT or SDEFAULTPRT defined, but not both.

Syntax



Parameters

`nnn%`
: Checks the capacity remaining in the GROUP when Telnet assigns an LU from that group. A message is issued when the specified percentage is reached. After the group exceeds the specified capacity, no other message is issued. After the capacity drops below 10 percent of the total capacity check amount, another capacity warning message is issued. Leave a blank space between DEFAULTLUS or SDEFAULTLUS and the comma (,) that is part of the capacity field.

`prt_name`
: The name of the printer LU.

`prt_name1..lu_name2`
: A range of printer LUs.

`range_rule`
: The wildcard method used for each character position.

Usage notes

See “Rules for LU name specification” on page 637 for LU name and LU range specification rules.

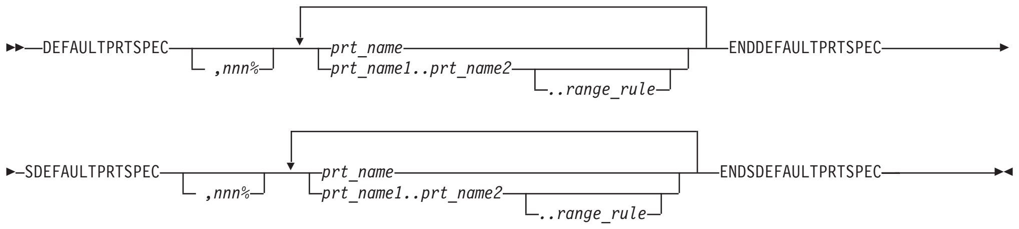
DEFAULTPRTSPEC or SDEFAULTPRTSPEC statement

Use the *optional* DEFAULTPRTSPEC or SDEFAULTPRTSPEC object and mapping statements to define a list or range of printer LUs with a default mapping to the NULL client identifier. This LU pool is used by a printer emulator requesting a specific connection if no other printer LU group maps specifically or generically to the client. The SDEFAULTPRTSPEC statement defines shared LU names rather than private ones.

Restrictions:

- This Telnet must have joined the XCF group to define shared objects. An active Telnet LU name server (LUNS) must exist in order for the profile to be processed and for the shared LUs to be usable.
- A profile can have either DEFAULTPRTSPEC or SDEFAULTPRTSPEC defined, but not both.

Syntax



Parameters

nnn%
: Checks the capacity remaining in the group when Telnet assigns an LU from that group. A message is issued when the specified percentage is reached. After the group exceeds the specified capacity, no other message is issued. After the capacity drops below 10 percent of the total capacity check amount, another capacity warning message is issued. Leave a blank space between DEFAULTLUS or SDEFAULTLUS and the comma (,) that is part of the capacity field.

prt_name
: The name of the printer LU.

prt_name1..lu_name2
: A range of printer LUs.

range_rule
: The wildcard method used for each character position.

Usage notes

See "Rules for LU name specification" on page 637 for LU name and LU range specification rules.

DESTIPGROUP statement

Use the *optional* DESTIPGROUP Client Identifier statement to define a group of destination IP addresses. The group name can be used on several mapping statements.

Syntax

```
                                      ┌─────────────────────────────────┐
                                      ↓                                 │
►►──DESTIPGROUP──DESTIP_group_name──┬─ip_addr─────────────────────────┬─┴──ENDDESTIPGROUP──────►◄
                                    ├─ipv4_subnet_mask──:──ipv4_subnet─┤
                                    ├─ipv6_subnet/prefix_len──────────┤
                                    └─ip_range1..ip_range2────────────┘
```

Parameters

DESTIP_group_name
The group name (up to 16 characters) that contains the destination IP addresses or subnets.

ipv4_subnet_mask:ipv4_subnet
An IPv4 format subnet. The ipv4_subnet_mask is a bit mask (expressed in dotted-decimal form) defining the subnetwork mask for a network route. The bits must be contiguous and start in the leftmost bit. The subnet_mask indicates the significant portion of the subnet. The subnet and an incoming IP address are each ANDed with the subnet_mask and then compared with each other to determine a match.

ipv6_subnet/prefix_len
An IPv6 format subnet. The prefix_len indicates how many significant bits there are starting from the leftmost bit. The subnet and an incoming IP address are each ANDed with the prefix_len number of bits and then compared with each other to determine a match.

ip_addr
The exact IP address of the destination host address that is the destination for a Telnet connection.

ip_range1..ip_range2
A range of IP addresses.

Restriction: Only the last octet of the IPv4 address and the last two hexadecimal bytes of the IPv6 address can be used as variables for the range.

Usage notes

- Any given IP address or combination of IP subnet mask and IP subnet can only appear once within all destination IP groups.
- The subnet and mask combination has no restrictions, including specific class address specifications.

HNGROUP statement

Use the *optional* HNGROUP Client Identifier statement to define a group of host names. The group name can be used on several mapping statements.

Tip: To cause the host name to be present in the control vector (CV64) information, or to make the host name that is used to inform the mechanism be associated with the CHECKCLIENTCONN statement, add a dummy HNGROUP-ENDHNGROUP statement block.

Syntax

```
                                 ┌────────┐
                                 ▼        │
►►──HNGROUP──hngroup_name──────hn_name────┴──ENDHNGROUP──────────────────────────►◄
```

Parameters

hngroup_name
The group name (up to 16 characters) that contains the host names.

hn_name
An exact, completely qualified host name or a wildcard host name.

Wildcards can be specified in two ways:

- Use a single asterisk (*) to indicate that any value is acceptable for a particular qualifier in a particular position within the host name. For example, *.*.IBM.COM matches USER1.RALEIGH.IBM.COM, but does not match USER1.TCP.RALEIGH.IBM.COM because this name includes an extra qualifier.

  Restriction: Use of a single asterisk cannot follow any non wildcarded name. For example, RALEIGH.*.COM is not allowed.
- Use a double asterisk (**) to indicate that any number of qualifiers are acceptable to the left of the asterisks. For example, **.IBM.COM matches USER1.IBM.COM, USER1.RALEIGH.IBM.COM, and USER1.TCP.RALEIGH.IBM.COM.

Both wildcard techniques require that the entire qualifier be wildcarded. For example, *USER.IBM.COM is not a valid use of a wildcard. In this case, use *.IBM.COM instead.

Usage notes

- Any given host name or wildcard host name can only appear one time within all HNGROUPs.
- Results in DNS hostname resolution for every new connection processed by the TN3270 server.
- See "Rules for host name specification" on page 639 for host name resolution and display information.

INTERPTCP statement

Use the *optional* INTERPTCP mapping statement to allow you to map a customized interpret table to a Client Identifier. This table is used to interpret incoming USS commands before the USS command processor is invoked. If the input string does not match any interpret table entry, the USS command processor parses the input string.

Syntax

```
►►──INTERPTCP──table_name──┬──────────────────────────────┬──────────────────────►◄
                           ├─Client_Identifier────────────┤
                           └─clid_type,Client_Identifier──┘
```

Parameters

table_name
The name of the interpret table load module.

clid_type
Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See “Rules for client identifier specification” on page 639 for details.

Client_Identifier
One of several Client Identifiers. See “Client identifier types and definitions” on page 638 for details. If no Client Identifier is specified, then it is considered the NULL Client Identifier.

Usage notes

- An assembled interpret table load module from VTAM can be used or one can be created. See z/OS Communications Server: IP Configuration Guide for coding details. Also see “Telnet INTERPRET table setup” on page 674.
- Always map a unique Client Identifier on each INTERPTCP statement. Otherwise, the last INTERPRET table mapping for the Client Identifier is used.
- The most common setup error is to fail to include the table load module in a load library accessible by TCP/IP.
- The INTERPRET table is used to check USS commands only. Therefore, INTERPRET table function is provided only for connections that are using a USS table.

IPGROUP statement

Use the *optional* IPGROUP Client Identifier statement to define a group of IP addresses. The group name can be used on several mapping statements.

Syntax

```
                           ┌──────────────────────────────────┐
                           ▼                                  │
►►──IPGROUP──ip_group_name───┬─ip_addr──────────────────────┬──┴──ENDIPGROUP──────►◄
                             ├─ipv4_subnet_mask──:──ipv4_subnet─┤
                             ├─ipv6_subnet/prefix_len───────┤
                             └─ip_range1..ip_range2─────────┘
```

Parameters

ip_group_name
The group name (up to 16 characters) that contains the Client IP addresses or subnets.

ipv4_subnet_mask:ipv4_subnet
An IPv4 format subnet. The ipv4_subnet_mask is a bit mask (expressed in dotted-decimal form) defining the subnetwork mask for a network route. The bits must be contiguous and start in the leftmost bit. The subnet_mask indicates the significant portion of the subnet. The subnet and an incoming IP address are each ANDed with the subnet_mask and then compared with each other to determine a match.

ipv6_subnet/prefix_len
An IPv6 format subnet. The prefix_len indicates how many significant bits there are starting from the leftmost bit. The subnet and an incoming IP address are each ANDed with the prefix_len number of bits and then compared with each other to determine a match.

ip_addr
The exact IP address of a particular client.

ip_range1..ip_range2
A range of IP addresses.

Restriction: Only the last octet of the IPv4 address and the last two hexadecimal bytes of the IPv6 address can be used as variables for the range.

Usage notes

- Any given client IP address can only appear one time within all IP Groups. A given combination of IP subnet mask and IP subnet can only appear once within all IP groups.
- The subnet and mask combination has no restrictions, including specific class address specifications.

LINEMODEAPPL statement

Use the *optional* LINEMODEAPPL mapping statement to map the initial application to be attempted when a Telnet client establishes a linemode connection.

Syntax

```
►►──LINEMODEAPPL──application_name──┬────────────────────────────────┬──────────►
                                    ├─Client_Identifier──────────────┤
                                    └─clid_type,Client_Identifier────┘

►──┬───────────┬──┬─────────┬──┬─────────┬─────────────────────────────────────►◄
   └─FIRSTONLY─┘  ├─LOGAPPL─┤  └─DEFONLY─┘
                  └─QINIT───┘
```

Parameters

application_name
The host application name, as specified in VTAMLST. The *application_name* can be network qualified in the format of a 1- to 8-character name of the network ID separated by a period (.), followed by a 1- to 8-character application name.

clid_type
Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See “Rules for client identifier specification” on page 639 for details.

Client_Identifier
One of several Client Identifiers. See “Client identifier types and definitions” on page 638 for details. If no client identifier is specified, then it is considered the NULL client identifier.

FIRSTONLY
When FIRSTONLY is specified, a solicitor or USSMSG10 screen is sent to the client after logoff from a default session when LUSESSIONPEND is coded. When FIRSTONLY is not specified, Telnet always requests a new session to the default application after logoff from the session when LUSESSIONPEND is coded. If LUSESSIONPEND is not coded the connection is dropped.

LOGAPPL
When LOGAPPL is specified, a session request to a host application that is not active is queued in VTAM instead of rejected. Telnet keeps the ACB open for the LU representing the client. When the application becomes active, VTAM initiates a session between the application and the Telnet LU.

QINIT
: Indicates that session requests should be queued, and when logging off the default application, Telnet should redrive the default application instead of issuing a USSMSG10 or Solicitor screen.

DEFONLY
: When DEFONLY is specified, the client is blocked from specifying any application name other than the one specified on the default application statement.

Usage notes

Always map a unique Client Identifier on each LINEMODEAPPL statement. Otherwise, the last LINEMODEAPPL mapping for the Client Identifier is used.

LINKGROUP statement

Use the *optional* LINKGROUP Client Identifier statement to define a group of link or interface names. The group name can be used on several mapping statements.

Syntax

```
                                       ┌──────────┐
                                       ▼          │
►►──LINKGROUP──linkgroup_group_name──────linkname─┴──ENDLINKGROUP──────────────────►◄
```

Parameters

linkgroup_group_name
: The group name (up to 16 characters) that contains the exact link or interface names or wildcard link or interface names.

linkname
: An exact link or interface or a wildcard link or interface name.

Linknames can be wildcarded when specified in a group.

- % or ? is a single-character position wildcard. It can be placed anywhere.
- * is a multi-position wildcard. It can only be placed at the end of the linkname.
- The two wildcard types can be used together. For example, L%%V5* is a valid wildcard name.

The position of the single wildcard (%) is used first to determine the most specific match. For example, the following wildcard names are checked in the order listed:

- C5CLINK*
- C5C%%%%*
- C5%LINK*
- C%CLINK*
- C%CLI%K*
- C%CLI%*
- C%CL%NK*
- C*

LUGROUP or SLUGROUP statement

Use the *optional* LUGROUP or SLUGROUP object statements to define a group of LUs. These group names can be used on the LUMAP statement to represent an LU pool. The SLUGROUP statement defines shared LU names rather than private ones.

Restrictions:

- This Telnet must have joined the XCF group to define shared LUs. An active Telnet LU name server (LUNS) must exist for the profile to be processed and for the shared LUs to be usable.
- All *lu_group_name* values on one profile must be unique, even though a profile can have both LU group names and shared LU group names defined.

Syntax

```
                                         ┌──────────────────────────────────────────┐
                                         ▼                                          │
►►──LUGROUP──lu_group_name──┬────────┬────┬─lu_name───────────────────────────┬─┴──────────►
                            ├─,nnn%──┤    └─lu_name1..lu_name2──┬──────────────┬┘
                            └─,EXIT──┘                          └─..range_rule─┘

►─ENDLUGROUP───────────────────────────────────────────────────────────────────────────────►


                                         ┌──────────────────────────────────────────┐
                                         ▼                                          │
►─SLUGROUP──lu_group_name──┬────────┬────┬─lu_name───────────────────────────┬─┴───────────►
                           └─,nnn%──┘    └─lu_name1..lu_name2──┬──────────────┬┘
                                                               └─..range_rule─┘

►─ENDSLUGROUP─────────────────────────────────────────────────────────────────────────────►◄
```

Parameters

`lu_group_name`
: The group name (1 - 8 characters in length) that contains the terminal LUs.

`nnn%`
: Checks the capacity remaining in the LUGROUP or SLUGROUP when Telnet assigns an LU from that group. A message is issued when the specified percentage is reached. After the group exceeds the specified capacity, no other message is issued. After the capacity drops below 10 percent of the capacity check amount, another capacity warning message is issued. Do not leave a blank space between the name and the comma that is part of the capacity field.

EXIT
: Indicates that the *lu_group_name* value is a user-written exit routine. When the LUGROUP statement is mapped to a Client Identifier, Telnet LU assignment invokes the exit routine to select an LU name. When the LU group is defined as an LU exit, the LU names or LU ranges are optional. When the names or ranges are provided, they act as seed values for the LU exit to use however it specifies. See "Telnet LU exit setup" on page 679 for exit details.

`lu_name`
: The name of the terminal LU.

lu_name1..lu_name2
A range of terminal LUs.

range_rule
The wildcard method used for each character position.

Tip: When practical, define LU ranges instead of long lists of LU names. LU name assignment from an LU range is more efficient than from a long list.

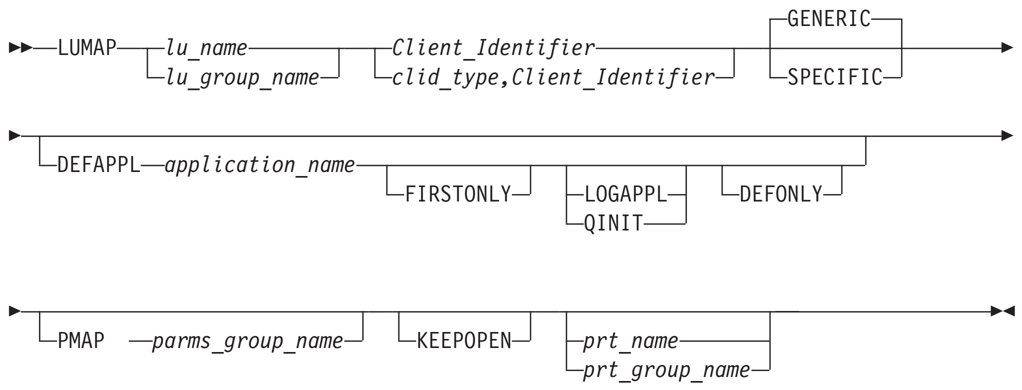
Usage notes

- See "Rules for LU name specification" on page 637 for LU name and LU Range specification rules.
- If the LU assigned to the connection is defined in LU groups mapped by both a LUG statement and an LUMAP statement, neither LU group can be defined as an LU exit.
- If a printer group is associated with the LU group on the LUMAP statement, the LU group cannot be defined as an LU exit.

LUMAP statement

Use the *optional* LUMAP mapping statement to define the mapping of an LU or group of LU objects to a Client Identifier.

Syntax



Parameters

lu_name
The name of the terminal LU.

lu_group_name
The group name that contains the terminal LUs.

clid_type
Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See "Rules for client identifier specification" on page 639 for details.

Client_Identifier
One of several client identifiers. See "Client identifier types and definitions" on page 638 for details.

GENERIC
Indicates that the LU or LUGROUP is checked for Generic connection requests.

Generic mapping statements also support Specific connection requests if there is no LU or LUGROUP mapped specifically to the client.

SPECIFIC
Indicates that the LU or LUGROUP is checked for Specific connection requests. Specific mapping statements are not used for Generic connection requests.

DEFAPPL *application_name*
Specifying DEFAPPL indicates the initial application to which Telnet connects. The *application_name* can be network qualified in the format of a 1- to 8-character name of the network separated by a period (.), followed by a 1- to 8-character application name.

FIRSTONLY
When FIRSTONLY is specified, a solicitor or USSMSG10 screen is sent to the client after logoff from a default session when LUSESSIONPEND is coded. When FIRSTONLY is not specified, Telnet always requests a new session to the default application after logoff from the session when LUSESSIONPEND is coded. If LUSESSIONPEND is not coded, the connection is dropped.

LOGAPPL
When LOGAPPL is specified, a session request to a host application that is not active is queued in VTAM instead of rejected. Telnet keeps the ACB open for the LU representing the client. When the application becomes active, VTAM initiates a session between the application and the Telnet LU.

QINIT
Indicates that session requests should be queued, and when logging off the default application, Telnet should redrive the default application instead of issuing a USSMSG10 or Solicitor screen.

DEFONLY
When DEFONLY is specified, the client is blocked from specifying any application name other than the one specified on the default application statement.

PMAP *parms_group_name*
Maps a ParmsGroup to an LU group. With this, parameters can be assigned based on the chosen LU name or group.

KEEPOPEN
Specifying KEEPOPEN means that all LUs identified in the lu_group_name or the LU identified by lu_name always have an OPEN ACB as long as the connection exists, whether or not a session exists. When KEEPOPEN is mapped to a connection, the MSG07 and LUSESSIONPEND functions are in effect whether or not they were explicitly coded.

prt_name
The name of an associated printer LU. Printer association requires a one-to-one match. A single LU name or an LUGROUP with a single LU must be specified when *prt_name* is used.

prt_group_name
The group that contains the printer LUs. Printer association requires a one-to-one match. The number of single names in the print group must equal the number of single names in the LUGROUP. The number of ranges and the number of LUs in each range must also match. The group cannot be defined as an LU exit.

Usage notes

- A single Client Identifier can have several LU names or LU groups mapped to it. See the LU assignment information in the Telnet topic in z/OS Communications Server: IP Configuration Guide for details.
- See "Rules for LU name specification" on page 637 for LU name specification rules.

MONITORGROUP statement

Use the *optional* MONITORGROUP statement to define parameters for monitoring the performance of connections mapped to this group.

Syntax

```
                                  ┌─Average───┐   ┌─Buckets───┐
►►──MONITORGROUP──mon_group_name──┼───────────┼───┼───────────┼──────────────────────►
                                  └─NoAverage─┘   └─NoBuckets─┘

   ┌─DynamicDR───┐   ┌─IncludeIP───┐   ┌─AvgSampPeriod 120──┐
►──┼─────────────┼───┼─────────────┼───┼────────────────────┼────────────────────────►
   └─NoDynamicDR─┘   └─NoIncludeIP─┘   └─AvgSampPeriod Ssec─┘

   ┌─AvgSampMultiplier 5────┐   ┌─Boundary1 50───┐   ┌─Boundary2 100──┐
►──┼────────────────────────┼───┼────────────────┼───┼────────────────┼──────────────►
   └─AvgSampMultiplier mult─┘   └─Boundary1 Bsec─┘   └─Boundary2 Bsec─┘

   ┌─Boundary3 200──┐   ┌─Boundary4 500──┐
►──┼────────────────┼───┼────────────────┼──ENDMONITORGROUP──────────────────────────►◄
   └─Boundary3 Bsec─┘   └─Boundary4 Bsec─┘
```

Parameters

mon_group_name
The name of the MonitorGroup.

DynamicDR/NoDynamicDR
Indicates whether or not Telnet should add the Definite Response (DR) request to the outbound TN3270E header if it was not set on by the application. If this option is not chosen, or the client does not support DR, Telnet uses a TIMEMARK to approximate the IP transit time.

IncludeIP/NoIncludeIP
Indicates whether or not Telnet should measure the transit time on the IP side of the connection.

Average/NoAverage
Indicates whether or not sliding averages should be calculated.

AvgSampPeriod Ssec
Specifies the sampling period for a sliding-window average. Default value of 120 seconds. The valid range is 1 - 99 999 999.

AvgSampMultiplier mult
Specifies the averaging period multiplier. Default value is 5. The valid range is 0 - 99 999 999.

Buckets/NoBuckets
Indicates whether or not time buckets are being used.

Boundary1 Bsec
Defines the upper boundary time, in milliseconds, for bucket 1 that contains the number of transactions whose transit times are greater than 0 and less than or equal to boundary1. The default value is 50 milliseconds. The valid range is 0 - 99 999 999.

Boundary2 Bsec
Defines the upper boundary time, in milliseconds, for bucket 2 that contains the number of transactions whose transit times are greater than boundary1 and less than or equal to boundary2. The default value is 100 milliseconds. The valid range is 0 - 99 999 999.

Boundary3 Bsec
Defines the upper boundary time, in milliseconds, for bucket 3 that contains the number of transactions whose transit times are greater than boundary2 and less than or equal to boundary3. The default value is 200 milliseconds. The valid range is 0 - 99 999 999.

Boundary4 Bsec
Defines the upper boundary time, in milliseconds, for bucket 4 that contains the number of transactions whose transit times are greater than boundary3 and less than or equal to boundary4. The default value is 500 milliseconds. The valid range is 0 - 99 999 999. Boundary4 also acts as the lower boundary for bucket 5, which has no upper boundary.

Usage notes

Each bucket maximum value must be higher than the preceding bucket maximum value. Zero can be specified in the first and subsequent buckets if those buckets are not wanted. After a positive value is specified, each succeeding bucket must have a higher value. A very large value, such as 99 999 990 can be used as an infinity value.

MONITORMAP statement

Use the *optional* MONITORMAP mapping statement to map a MONITORGROUP to a Client Identifier.

Syntax

```
►►──MONITORMAP──mon_group_name──┬─Client_Identifier───────────┬──────────────►◄
                                └─clid_type,Client_Identifier─┘
```

Parameters

mon_group_name
The name of the MONITORGROUP.

clid_type
Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See "Rules for client identifier specification" on page 639 for details.

Client_Identifier
One of several client identifiers. See "Client identifier types and definitions" on page 638 for details.

Usage notes

See the connection monitoring mapping information in the z/OS Communications Server: IP Configuration Guide.

PARMSGROUP statement

Use the *optional* PARMSGROUP Object statement to define parameters that are mapped to a subset of all clients. The PARMSGROUP statements mapped to a client override those defined in the TELNETGLOBALS, TELNETPARMS, or BEGINVTAM block.

Syntax

```
                                      ┌──────────────┐
                                      ▼              │
►►──PARMSGROUP──parmsgroup_name────valid_stmts──┴──ENDPARMSGROUP──────────────────►◄
```

Parameters

parmsgroup_name
: The group name (up to eight characters) that contains the Telnet parameter statements.

valid_stmts
: Any Telnet statement that is permitted in PARMSGROUP. See Table 33 on page 590 for a list of valid statements.

Usage notes

Security parameters are accepted for ports defined as SECUREPORT ports. See "Rules for Telnet parameter statements and security parameters" on page 593 for details.

PARMSMAP statement

Use the *optional* PARMSMAP mapping statement to map a PARMSGROUP to a Client Identifier.

Syntax

```
►►──PARMSMAP──parmsgroup_name──┬─Client_Identifier──────────┬──────────────────►◄
                               └─clid_type,Client_Identifier─┘
```

Parameters

parmsgroup_name
: The name of the PARMSGROUP.

clid_type
: Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See "Rules for client identifier specification" on page 639 for details.

Client_Identifier
: One of several client identifiers. See "Client identifier types and definitions" on page 638 for details.

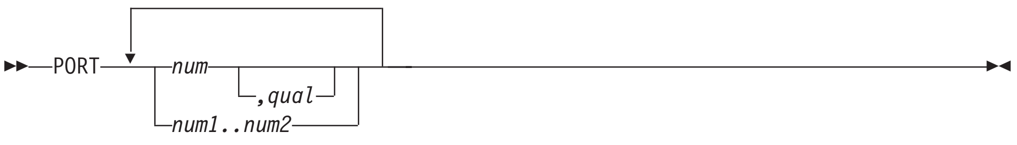
Usage notes

A single Client Identifier can have several PARMSGROUPS mapped to it. See the PARMSGROUP assignment information in the Telnet topic in z/OS Communications Server: IP Configuration Guide for details.

PORT statement

Use the *optional* PORT statement to associate the BEGINVTAM block with the correct TELNETPARMS block when multiple ports are used.

Syntax



Parameters

num
A specified port number.

,qual
Qualifies the PORT address with a destination IP address or with a specific link or interface name.

num1..num2
A consecutive range of ports starting with *num1* and ending with *num2*. *num2* must be greater than *num1*.

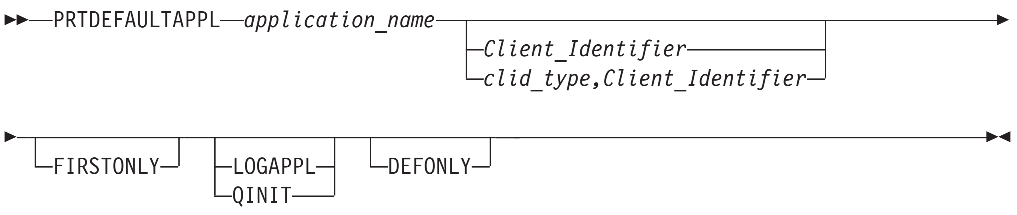
Usage notes

- If port,qual is coded, it must match the qualifier used in the PORT, SECUREPORT, or TTLSPORT statement in the TELNETPARMS block.
- The PORT statement must be the first statement following the BEGINVTAM statement.

PRTDEFAULTAPPL statement

Use the *optional* PRTDEFAULTAPPL mapping statement to map the initial application to be tried when a Telnet client establishes a printer connection. The application can be a particular VTAM application, such as CICS.

Syntax



Parameters

`application_name`
: The host application name, as specified in VTAMLST. The *application_name* can be network qualified in the format of a 1- to 8-character name of the network ID separated by a period (.), followed by a 1- to 8-character application name.

`clid_type`
: Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See "Rules for client identifier specification" on page 639 for details.

`Client_Identifier`
: One of several Client Identifiers. See "Client identifier types and definitions" on page 638 for details. If no Client Identifier is specified, then it is considered the NULL Client Identifier.

`FIRSTONLY`
: When FIRSTONLY is specified, the printer LU remains active with an open ACB after initial session logoff. When FIRSTONLY is not specified, Telnet always requests a new session to the default application after logoff from the session when LUSESSIONPEND is coded. If LUSESSIONPEND is not coded, the connection is dropped.

`LOGAPPL`
: When LOGAPPL is specified, a session request to a host application that is not active is queued in VTAM instead of rejected. Telnet keeps the ACB open for the LU representing the client. When the application becomes active, VTAM initiates a session between the application and the Telnet LU.

`QINIT`
: Indicates that session requests should be queued, and when logging off the default application, Telnet should redrive the default application instead of issuing a USSMSG10 or Solicitor screen.

`DEFONLY`
: When DEFONLY is specified, the client is blocked from specifying any application name other than the one specified on the default application statement.

Usage notes

Always map a unique Client Identifier on each PRTDEFAULTAPPL statement. Otherwise, the last PRTDEFAULTAPPL mapping for the Client Identifier is used.

PRTGROUP or SPRTGROUP statement

Use the *optional* PRTGROUP or SPRTGROUP object statements to define a group of printer LUs. These group names can be used on the PRTMAP statement to represent a printer pool. The SPRTGROUP statement defines shared LU names rather than private ones.

Restrictions:

- This Telnet must have joined the XCF group to define shared printer LUs. An active Telnet printer LU name server (LUNS) must exist for the profile to be processed and for the shared printer LUs to be usable.
- All *prt_group_name* values on one profile must be unique, even though a profile can have both PRTGROUPs and SPRTGROUPs defined.

Syntax

```
                                          ┌──────────────────────────────────────────┐
                                          ▼                                          │
►►──PRTGROUP──prt_group_name──┬────────┬──┬─prt_name─────────────────────────────┬──┴──────►
                              ├─,nnn%──┤  └─prt_name1..prt_name2──┬──────────────┤
                              └─,EXIT──┘                          └─..range_rule─┘

►──ENDPRTGROUP─────────────────────────────────────────────────────────────────────────────►


                                          ┌──────────────────────────────────────────┐
                                          ▼                                          │
►──SPRTGROUP──prt_group_name──┬────────┬──┬─prt_name─────────────────────────────┬──┴──────►
                              └─,nnn%──┘  └─prt_name1..prt_name2──┬──────────────┤
                                                                  └─..range_rule─┘

►──ENDSPRTGROUP───────────────────────────────────────────────────────────────────────────►◄
```

Parameters

prt_group_name
The group name (1 - 8 characters in length) that contains the printer LUs.

prt_name
The name of the printer LU.

nnn%
Checks the capacity remaining in the PRTGROUP or SPRTGROUP when Telnet assigns an LU from that group. A message is issued when the specified percentage is reached. After the group exceeds the specified capacity, no other message is issued. After the capacity drops below 10 percent of the total capacity check amount, another capacity warning message is issued. Do not leave a blank space between the name and the comma that is part of the capacity field.

EXIT
Indicates that the *prt_group_name* value is a user-written exit routine. When the PRTGROUP statement is mapped to a Client Identifier, Telnet LU assignment invokes the exit routine to select an LU name. When the LU group is defined as an LU exit, the LU names or LU ranges are optional. When the names or ranges are provided, they act as seed values for the LU exit to use however it specifies. See "Telnet LU exit setup" on page 679 for exit details.

prt_name1..prt_name2
A range of printer LUs.

range_rule
The wildcard method used for each character position.

Tip: When practical, define LU ranges instead of long lists of LU names. LU name assignment from an LU range is more efficient than from a long list.

Usage notes

- See "Rules for LU name specification" on page 637 for LU name and LU range specification rules.
- If the LU assigned to the connection is defined in LU groups mapped by both a LUG statement and an PRTMAP statement, neither LU group can be defined as an LU exit.

- If the printer LU group is used as an associated printer group on an LUMAP statement, the group cannot be defined as an LU exit.

PRTMAP statement

Use the *optional* PRTMAP mapping statement to define the mapping of a printer LU or group of printer LUs objects to a client identifier.

Syntax

```
                                                                  ┌─GENERIC──┐
►►──PRTMAP──┬─prt_name───────┬──┬─Client_Identifier───────────┬──┼──────────┼────────►
            └─prt_group_name─┘  └─clid_type,Client_Identifier─┘  └─SPECIFIC─┘

►─┬──────────────────────────────────────────────────────────────────────────┬──────►
  └─DEFAPPL──application_name──┬───────────┬──┬─────────┬──┬─────────┬──┘
                               └─FIRSTONLY─┘  ├─LOGAPPL─┤  └─DEFONLY─┘
                                              └─QINIT───┘

                                  ┌─KEEPOPEN─┐
►─┬──────────────────────────┬──┴──────────┴─────────────────────────────────────►◄
  └─PMAP ──parms_group_name─┘
```

Parameters

prt_name
: The name of the printer LU.

prt_group_name
: The group name that contains the printer LUs.

clid_type
: Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See “Rules for client identifier specification” on page 639 for details.

Client_Identifier
: One of several client identifiers. See “Client identifier types and definitions” on page 638 for details.

GENERIC
: Indicates that the LU or PRTGROUP are checked for Generic connection requests. Generic mapping statements also support Specific connection requests if there is no LU or PRTGROUP mapped specifically to the client.

SPECIFIC
: Indicates that the LU or PRTGROUP are checked for Specific connection requests. Specific mapping statements are not used for Generic connection requests.

DEFAPPL *application_name*
: Specifying DEFAPPL indicates the initial application to which Telnet connects. The *application_name* can be network qualified in the format of a 1- to 8-character name of the network separated by a period (.), followed by a 1- to 8-character application name.

FIRSTONLY
: When FIRSTONLY is specified, the printer LU remains active with an open ACB after initial session logoff from the default session. When FIRSTONLY is not specified, Telnet always requests a new session to the default application

after logoff from the session when LUSESSIONPEND is coded. If LUSESSIONPEND is not coded, the connection is dropped.

LOGAPPL
When LOGAPPL is specified, a session request to a host application that is not active is queued in VTAM instead of rejected. Telnet keeps the ACB open for the LU representing the client. When the application becomes active, VTAM initiates a session between the application and the Telnet LU.

QINIT
Indicates that session requests should be queued, and when logging off the default application, Telnet should redrive the default application instead of issuing a USSMSG10 or Solicitor screen.

DEFONLY
When DEFONLY is specified, the client is blocked from specifying any application name other than the one specified on the default application statement.

PMAP *parms_group_name*
Maps a ParmsGroup to an LU group. With this, parameters can be assigned based on the chosen LU name or group.

KEEPOPEN
Specifying KEEPOPEN means that all LUs identified in the lu_group_name or the LU identified by lu_name always have an OPEN ACB as long as the connection exists, whether or not a session exists. For printers, this option is always set. When KEEPOPEN is mapped to a connection, the MSG07 and LUSESSIONPEND functions are in effect whether or not they were explicitly coded.

Usage notes

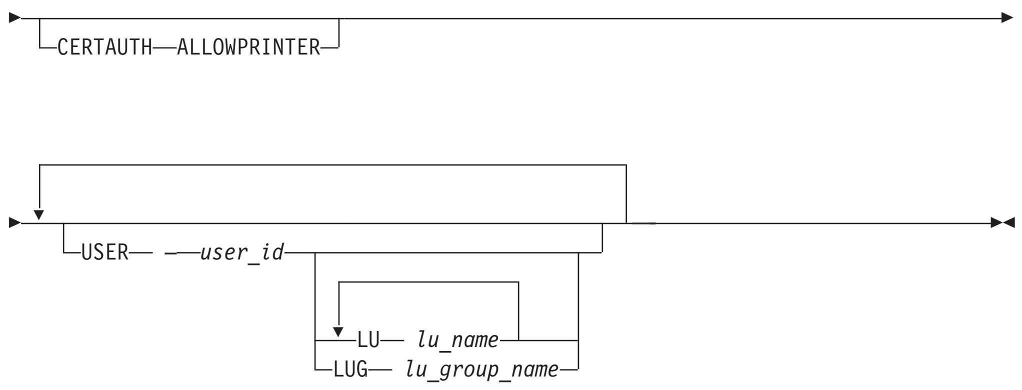
- A single Client Identifier can have several printer LU names or printer LU groups mapped to it. See the LU assignment information in the Telnet topic of the z/OS Communications Server: IP Configuration Guide.
- See "Rules for LU name specification" on page 637 for LU name specification rules.

RESTRICTAPPL statement

Use the *optional* RESTRICTAPPL mapping and security statement to restrict access to the specified application. This statement should be followed by user parameters defining each user who is authorized to use the application. Users are prompted to identify themselves with a password. RACF or an equivalent security program is used to validate the password. If no user parameters are specified, the application cannot be accessed.

Syntax

```
►►──RESTRICTAPPL──application_name──┬────────────────┬──QSESSion──┬──────┬──────────►
                                    └─DISCONNECTABLE─┘            └─,sec─┘
```



Parameters

application_name
The host application name, as specified in VTAMLST.

Single-character position wildcards (%) are permitted anywhere in the application name and the multi-character wildcard (*) is permitted at the end of an application name. For example, A%CICS* restricts connections to A1CICS01, A1CICS02, ABCICS4A, and so on. A single * restricts all applications.

DISCONNECTABLE
When DISCONNECTABLE is specified, VTAM notifies the application to disconnect, rather than log off a user, when the session is dropped.

QSESSion
Indicates this application queues a session request when passing the session to another primary application. When Telnet receives an UNBIND of the new session, Telnet waits for a BIND to reestablish the original queued session.

sec
When QSESSion is coded, this value determines the number of seconds Telnet waits before checking whether a BIND was received. The range is 1 - 99999999. If no BIND is received in the time specified, Telnet stops waiting and continues cleaning up the connection as if QSESSion had not been coded. There is no default value. If *sec* is not coded, the connection never checks whether a BIND is received. Telnet waits until a BIND is received or the connection is dropped.

CERTAUTH
Specifies to use the derived User ID based on the SSL Client Certificate (enhanced LU mapping support for dynamic IP environments) and skips the Restrictappl password validation process. If Express Logon is being used, the User ID returned from security lookup for the latest Client Certificate/Applid combination is used. If not using Express Logon, the User ID returned at initial connection time from security lookup for just the Client Certificate is used.

ALLOWPRINTER
Specifies that any printer connection matching this RESTRICTAPPL statement is treated as if it matched an ALLOWAPPL statement. No user ID or password is requested. Printer emulators do not support user ID and password requests. The ALLOWPRINTER parameter gives you the ability to have terminal connections and printer connections mapped on a single RESTRICTAPPL statement. However, the printer connections exist at the lower security level that is provided by the ALLOWAPPL statement.

USER *user_id*
: The user ID, one to eight characters long. Single-character wildcards (%) are permitted anywhere in the user name and the multi-character wildcard (*) is permitted at the end of the user name. A single * allows all users.

LU *LU_name*
: The logical name of the Telnet terminal LU. This parameter allows you to optionally specify which terminal LUs can be used to establish a session with the named VTAM host application.

LUG *LU_group_name*
: The name of an LUGROUP or PRTGROUP. This option allows you to specify an LUGROUP or PRTGROUP, where any LU in the group can be used to establish a session with the named VTAM host application. If the same name defines both an LUGROUP and a PRTGROUP, the LUGROUP is used. The group can be a new group consisting of a combination of names or range list names from existing LUGROUPs and PRTGROUPs. This allows both terminals and printers to be on the same RESTRICTAPPL-USER statement.

Usage notes

- LU and LUG keywords are mutually exclusive. If both are specified in any order, only the LUG is processed. If multiple LUG keywords are specified, only the last is accepted and processed.
- Applications that do CLSDST Pass also require a RESTRICTAPPL or ALLOWAPPL statement for the target application.
- If the LU assigned to the connection is defined in LU groups mapped by both a LUG statement and an LUMAP/PRTMAP statement, neither LU group can be defined as an LU exit.

USERGROUP statement

Use the *optional* USERGROUP object statement to define a group of user IDs. The group name can be used on several mapping statements.

Syntax

```
                                   ┌──────────┐
                                   ▼          │
►►──USERGROUP──user_group_name────user_IDs───┴──ENDUSERGROUP──────────────────────►◄
```

Parameters

user_group_name
: The group name (up to 16 characters) that contains user ID names which represent clients when the client certificate is translated into a user ID.

user_IDs
: An exact user ID name or a wildcard user ID name.

  User ID names can be wildcarded when specified in a group.
  - % or ? is a single character position wildcard. It can be placed anywhere.
  - * is a multi-position wildcard. It can only be placed at the end of the user ID.
  - The two wildcard types can be used together. For example, U%%V5* is a valid wildcard name.

The position of the single wildcard (%) is used first to determine the most specific match. For example, the following wildcard names are checked in the order listed:

- M5MUSER*
- M5M%%%%*
- M5%USER*
- M%MUSER*
- M%MUS%R*
- M%MUS%*
- M%MU%ER*
- M*

USSTCP statement

Use the *optional* USSTCP mapping statement to map a customized USS table to a Client Identifier. You can use an existing table or create a USS table, assemble it, and load it into your system library.

Syntax

```
►►──USSTCP──┬─3270_table_name────────────────┬──┬──────────────────────────────┬──────────►◄
            └─3270_table_name,scs_table_name─┘  ├─Client_Identifier────────────┤
                                                └─clid_type,Client_Identifier──┘
```

Parameters

3270_table_name
The name of the 3270 format USS table load module.

scs_table_name
The name of the SCS format USS table load module.

clid_type
Specifies the type of Client Identifier. It is required if USERID or DESTIP are specified. See “Rules for client identifier specification” on page 639 for details.

Client_Identifier
One of several client identifiers. See “Client identifier types and definitions” on page 638 for details. If no Client Identifier is specified, then it is considered the NULL Client Identifier.

Usage notes

- An assembled USS table load module from VTAM can be used or one can be created. For coding details, see z/OS Communications Server: IP Configuration Guide. Also see “Telnet USS table setup” on page 665.
- Always map a unique Client Identifier on each USSTCP statement. Otherwise, the last USS table mapping for the Client Identifier is used.
- The most common setup error is to fail to include the table load module in a load library accessible by TCP/IP.
- If a default application and a USS table are both mapped to the same Client Identifier, the default application is used. The USS messages are used in case of an error or if FIRSTONLY is specified on DEFAULTAPPL.
- If an SCS format USS table is specified, it is used for all TN3270E connections. Non-TN3270E connections continue to use the 3270 format USS table. If no SCS

format USS table is specified, all connections use the 3270 format USS table. In this case, a BIND/UNBIND is sent to the TN3270E client before/after USS processing.

Telnet USS table setup

This topic includes information about the Telnet USS table setup, including general rules and macroinstructions.

USSCMD
: The USSCMD macroinstruction is used to define Telnet terminal operator commands.

USSMSG
: The USSMSG macroinstruction defines Telnet terminal operator messages (USSMSGxx).

USSPARM
: The USSPARM macroinstruction defines an operand or positional parameter that can be specified on a command identified by the USSCMD macroinstruction. It also defines default values for the operand or positional parameter.

  There can be multiple USSPARM macroinstructions associated with a USSCMD macroinstruction. For each operand (keyword or positional), code a USSPARM macroinstruction.

USSEND
: The USSEND macroinstruction delimits the end of the USS table.

USSTAB
: The USSTAB macroinstruction indicates the beginning of a USS table.

General usage rules for Telnet USS macroinstructions

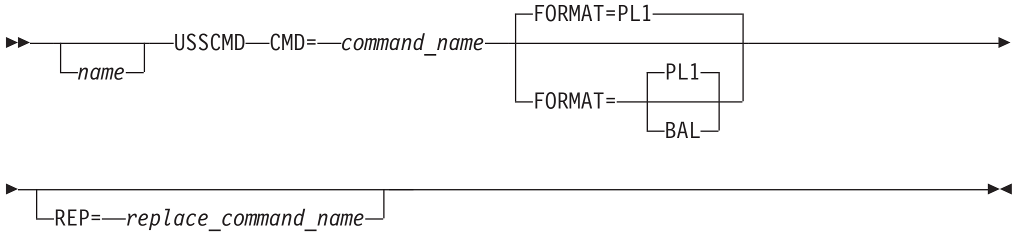
Observe the following general usage rules for Telnet USS macroinstructions:

- The Telnet USS macroinstructions can be coded exactly as the VTAM macroinstructions. A few VTAM parameters are not supported by Telnet. In these cases, the parameter value is ignored and does not interfere with the execution of the macroinstruction. Differences between Telnet and VTAM are listed under usage notes for each macroinstruction.
- An assembled and linked VTAM USS table can be used directly by Telnet. Unsupported statements are ignored and do not interfere with the processing of the command.
- For additional information about installing or changing an interpret table, See the z/OS Communications Server: SNA Resource Definition Reference, which contains instructions for using the Telnet solicitor or USS Logon Panel.
- A sample USS table is located in SEZAINST(EZBTPUST).
- The USS Macroinstructions can be found in *hlq*.SISTMAC1, the VTAM macro library.

USSCMD macroinstruction

Use the USSCMD macroinstruction to define a Telnet operator or terminal operator command.

Syntax

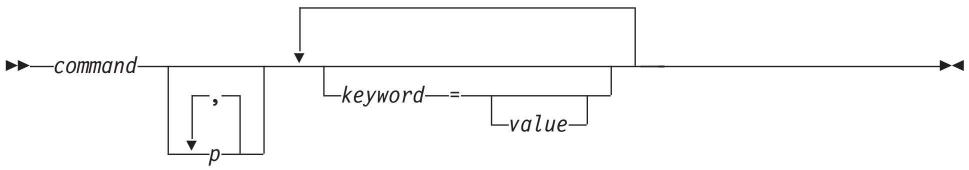


Parameters

name
: Specifies the name assigned to the macroinstruction.

CMD=*command_name*
: Specifies the command name assigned to the macroinstruction.

FORMAT=BAL
: Specifies the user-defined command indicated on this USSCMD macroinstruction in Basic Assembler Language (BAL) syntax.



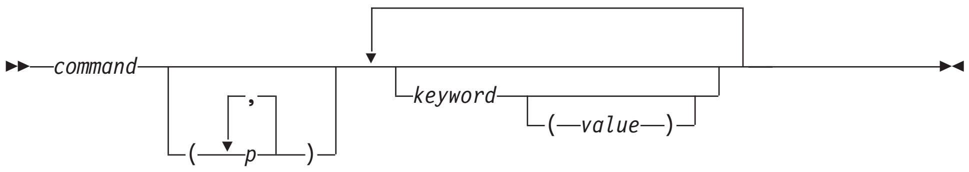
command
: Identifies the command. It is followed by one or more blanks.

p
: Specifies one or more positional operands. Positional operands are entered in the format P*n*, where *n* is the position number of the operand. Each operand (unless it is the last in the command) is followed by a comma. Positional operands must appear before any keyword operands.

keyword
: Specifies keyword operand associated with the command. Each operand (unless it is the last in a command) is followed by a comma.

value
: Determines the value assigned to a keyword operand.

FORMAT=PL1
: Specifies the user-defined command specified on this USSCMD macroinstruction in PL/I programming syntax.



command
: Identifies the command. It is followed by one or more blanks or by a left parenthesis (that is, positional operands).

p
: Specifies one or more positional operands. Positional operands are entered in the format P*n*, where *n* is the position number. If positional operands are used, the parentheses must be coded.

keyword
: Used to enter each operand parameter. Each operand must be followed by one or more blanks or by a value enclosed in parentheses.

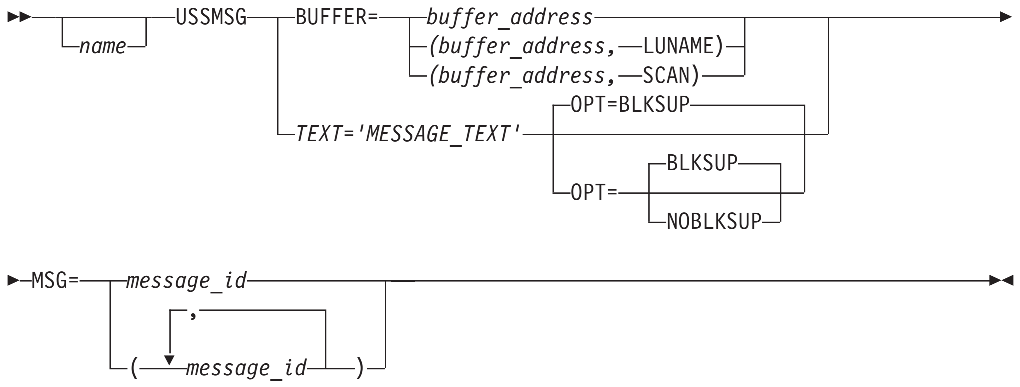
value
: The value assigned to a keyword operand.

REP=*replace_command_name*
: Specifies the valid command that is to replace the user-defined command indicated by the CMD operand. If the REP operand is not coded, the value specified in the CMD operator in used.

USSMSG macroinstruction

Use the USSMSG macroinstruction to define Telnet terminal operator messages (USSMSGxx).

Syntax



Parameters

name
: Specifies the name assigned to the macroinstruction.

buffer_address
: Specifies the address (name) of an area of storage defined to contain the message text and a header indicating the length of the message text. The storage area must be formatted as shown in Figure 26 on page 668.

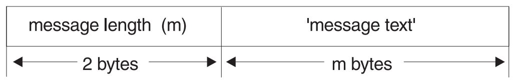


Figure 26. USS message layout in storage

The message text defined in the storage area must follow the USSEND macroinstruction.

The message text is sent to the terminal operator as it appears in the storage area. Telnet does not modify or translate the message text. You are responsible for including any device-dependent control characters within the message. The data format must be 3270 data stream or SNA character stream (SCS). Both are not supported by Telnet.

LUNAME|SCAN
Specifies that the character strings listed in Table 39 are replaced with the appropriate values in the position in the message where the character string occurred. The entire string specified by BUFFER is searched, using the character @. System symbolics are also replaced with their appropriate value. When using the system symbolics in the USS table, an extra ampersand (&) must be prepended to the system symbolic for the assembler compiler to create the correct output. For example, system symbolic &sysname. must be in the table as '&&sysname. for the compiled output to be '&sysname.'

Table 39. Variables substituted for USSMSG

Character string	Message text	Format
@@@@DATE	Current Date	8 bytes, in the format specified by the DATEFRM and DATEDLM operands on the USSTAB macroinstruction.
@@@@@@@@@IPADDR @..@IPADDR(1)	Client IP Address	15 bytes, leading 0's suppressed, left-aligned, with trailing blanks if needed.
@...@IPHOSTNAME (2)	Client host name	40 bytes, name left-aligned with trailing blanks if needed.
@@LUNAME (3)	Client LU Name (SLU)	8 bytes, name left-aligned with trailing blanks if needed.
@@PRT	Client Port Address	5 bytes and leading 0's are not suppressed.
@@@@RUNAME	Failing operation Name	10 bytes, name left-aligned with trailing blanks if needed.
@@@SENSE	Sense Code or Return Code	8 bytes.
@@@@TIME	Current Time	8 bytes in the HH_MM_SS format, where an underscore (_) is the delimiter specified on the TIMEDLM operand of the USSTAB macroinstruction.

Table 39. Variables substituted for USSMSG (continued)

Character string	Message text	Format
@HOSTNET @@@NETID @...@NQN (4) @@SSCPNM @@@@@@@@@@ZONEID		Placeholders for Telnet. Accepted for use, but are set to blanks.

Notes:
1. IPv6 IPADDR must be preceded by 33 @ symbols.
2. IPHOSTNAME must be preceded by 30 @ symbols.
3. @@LUNAME is substituted when it is known. For TN3270 connections, the LU name is not known until after the MSG10 screen is sent to the end-user because the application name is not yet known.
4. NQN must be preceded by 14 @ symbols.

message_id

Specifies which message or messages are defined by this macroinstruction. Table 40 shows the default table variable substitution and examples.

For terminal operator messages, enter decimal integers in the range 0 - 14. The numbers 0 - 14 correspond to the USS messages with message IDs of USSMSG00 through USSMSG14, respectively.

Restriction: USSMSG00 is not defined in the IBM-supplied USS table. If you do not define this message, no message is sent in this case.

Table 40. Default table variable substitution

Message	Variable	Example
MSG00	Command	% COMMAND ACCEPTED
MSG01	Command	INVALID % COMMAND SYNTAX
MSG02	Command	% COMMAND UNRECOGNIZED
MSG03	Command parameter	% PARAMETER EXTRANEOUS
MSG04	• Command parameter • Command parameter value	% PARAMETER VALUE %(2) NOT VALID
MSG05	None	UNSUPPORTED FUNCTION
MSG06	Message not used	Not applicable — NOT USED BY TELNET
MSG07	• LU name • Operation that failed • Sense Code 3 or Return Code. See message EZZ6035I for return code explanation.	%(1) UNABLE TO ESTABLISH SESSION — %(2) FAILED WITH SENSE %(3)
MSG08	None	INSUFFICIENT STORAGE
MSG09	Message not used	Not applicable — NOT USED BY TELNET
MSG10	None	A 3270 data format screen
MSG11	Message not used	Not applicable — NOT USED BY TELNET

Table 40. Default table variable substitution (continued)

Message	Variable	Example
MSG12	None	REQUIRED PARAMETER OMITTED
MSG13	Text after IBMTEST echoed back	IBMECHO %
MSG14	Message number that could not be displayed	USS MESSAGE % NOT DEFINED

OPT=BLKSUP|NOBLKSUP
BLKSUP specifies that extraneous blanks are suppressed from the message. Any sequence of two or more blanks is converted into a single blank. NOBLKSUP specifies that extraneous blanks are not suppressed from the message. Any sequence of two or more blanks is presented unchanged in the message.

message_text

Specifies the text to use in the USS messages identified by the MSG operand. Within *message_text*, place any combination of the character strings described in Table 39 on page 668. Telnet places the strings with the values shown in the table.

Rule: Blank suppression always occurs, even if OPT=NOBLKSUP is coded.

Usage notes

For TN3270E, this limitation exists. Unless specific IP-to-LU mapping is used, the LU name is not known for non-TN3270E sessions until an application is chosen from the MSG10 screen. Therefore, no @@LUNAME substitution takes place on the MSG10 screen for non-TN3270E sessions.

USSPARM macroinstruction

Use the USSPARM macroinstruction to define an operand or positional parameter that can be specified on a command identified by the USSCMD macroinstruction. It also defines values for the operand or positional parameter. There can be multiple USSPARM macroinstructions associated with a USSCMD macroinstruction. For each operand (keyword and positional), code a USSPARM macroinstruction.

Syntax

```
►►──┬──────┬──USSPARM──PARM=──┬─parm_operand_name─┬──────────────────────────►
    └─name─┘                  └─P_number──────────┘

►──┬───────────────────────┬──┬───────────────────────┬─────────────────────►
   └─DEFAULT=──default_value─┘  └─REP=──rep_operand_name─┘
   ┌─TRANSLATE=YES──────────┐
►──┼────────────────────────┼──┬─────────────────────────┬───────────────────►◄
   │              ┌─YES─┐   │  └─VALUE=──value_value─────┘
   └─TRANSLATE=───┤     ├───┘
                  └─NO──┘
```

Parameters

name
: Specifies the name assigned to the macroinstruction.

parm_operand_name
: Specifies the keyword parameter in the user-entered command to which this USSPARM macroinstruction applies. *parm_operand_name* must be 1–8 alphanumeric characters.

P_number
: Specifies a positional parameter, where *number* is a decimal integer from 1 to the maximum number of positional parameters for the command. *P_number* indicates the positional parameter in the user-entered command to which this USSPARM macroinstruction applies.

default_value
: Specifies a default value to be used if the operand is omitted when the command is entered. If DEFAULT is not specified, the operand is treated as if it were not entered.

  If the parameter in the PARM operand allows a network-qualified name to be specified, then the value of DEFAULT can be a network-qualified name.

rep_operand_name
: Specifies the parameter is replaced with *rep_operand_name*. The value for *rep_operand_name* must be 1–8 alphanumeric characters. The value of the operand is assigned from the parameter specified by PARM. If PARM specifies a keyword parameter, its value is assigned to the operand specified by REP. If PARM specifies a positional parameter, its value is treated as if it were an operand value and it is assigned to the operand specified by REP.

  If REP is not coded, it takes the value of PARM. (That is, the user-entered parameter is used as entered.)

  Positional parameters such as P1 and P2 can also be used as operands.

TRANSLATE=YES|NO
: Controls translation of the specified USSPARM.

  TRANSLATE=YES is the default and specifies that the USSPARM is translated using the translation table associated with the USS table this USSPARM is coded in, unless the character string is within single quotation marks. Character strings within single quotation marks are not translated.

  TRANSLATE=NO specifies that the USSPARM is not translated. TRANSLATE=NO is intended to be coded only on the USSPARM for DATA when the data contains a mixed-case password and the destination application supports mixed-case passwords. For more information about mixed-case passwords, see z/OS Communications Server: IP Configuration Guide.

value_value
: Specifies the default value to be used if the operand specified by the PARM operand is entered without a value.

VALUE is in contrast with the DEFAULT operand, which specifies the default to be used if the operand itself is not entered.

If multiple VALUE operands are specified for the same operand, the first VALUE operand is used.

If the parameter in the PARM operand allows a network-qualified name to be specified, then the value of VALUE can be a network-qualified name.

Examples

The following code is an example using TRANSLATE=NO to bypass the translation table and pass a mixed case user ID and password, assuming the translation table is used to convert all text to upper case.

```
AUSSTAB  USSTAB
APPL1    USSCMD  CMD=APPL1,REP=LOGON,FORMAT=PL1
         USSPARM PARM=APPLID,REP=APPLID,DEFAULT=APPL1
         USSPARM PARM=P1,REP=DATA,TRANSLATE=NO
         USSPARM PARM=P2,REP=LOGMODE
         USSEND
```

The following terminal operator command is entered:

```
appl1 user1/PaSsWrD1 interact
```

The command sends DATA() and a LOGMODE() to application APPL1. The LOGMODE value is translated to upper case. No character translation was performed on user1/PaSsWrD1 because TRANSLATE=NO was coded on the DATA USSPARM. The application receives a logmode value of INTERACT and data value of user1/PaSsWrD1.

Usage notes

- The DEFAULT and VALUE operands cannot be coded on the same USSPARM macroinstruction. To use both operands, code two USSPARM macroinstructions with the same value specified for PARM. The macroinstruction specifying VALUE must precede the one containing the DEFAULT operand. If REP is to be specified, it must be on the macroinstruction containing the VALUE operand. For example,

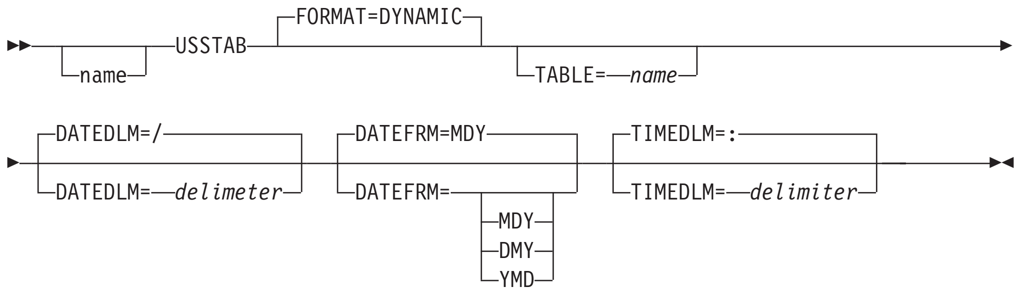
  ```
  USSPARM P=T,REP=TYPE,VALUE=COND
  USSPARM P=T,REP=TYPE,DEFAULT=COND
  ```
- For multiple specifications of the same parameter, the last value specified is used. An exception is if positional parameters are used to represent the DATA parameter. Specifying multiple data positional parameters permits a data string with a blank to be entered. Each blank acts as a parameter delimiter. If the number of blanks is known, multiple DATA parameters can be used instead of using an interpret table. For example, a LOGON TSO command can have two DATA parameters. The first could be USERID and the second could be the PROC. Telnet accepts both parameters and passes both as data to the host application with a blank between the parameters.
- Parameters used by Telnet are:

  ```
  LOGON APPLID,LOGMODE,DATA
  LOGOFF
  IBMTEST # of retries
  ```

USSTAB macroinstruction

Use the USSTAB macroinstruction to indicate the beginning of a USS table.

Syntax



Parameters

name
Specifies the required CSECT name for the USS table.

FORMAT= DYNAMIC
Specifies how the USS table is formatted. Dynamic is required for Telnet.

TABLE=*name*
Specifies the translation table that is used by Telnet to translate character-coded commands. If a translation table is coded in the specified USS table, the table that is used. If no table is coded, the table in the IBM default EZBTPUST is used. If EZBTPUST has been altered and no longer contains a translation table, an internal translation table is used that is the same as the table in EZBTPUST.

DATEDLM
Specifies the character to be used as a delimeter to separate the month, day, and year parts of the date where @@@@DATE is specified in the message text. The slash (/) is used if DATEDLM is not specified. An ampersand (&) and single quotation mark (') are not valid delimiters.

DATEFRM
Specifies the date format to be used where @@@@DATE is specified in the message text. Note that the delimiter used between the month, day, and year is specified on the DATEDLM operand.

DMY Specifies the day, followed by month, followed by year as dd_mm_yy, where an underscore (_) is the delimiter specified on the DATEDLM operand.

MDY Specifies the month, followed by day, followed by year as mm_dd_yy, where an underscore (_) is the delimiter specified on the DATEDLM operand.

YMD Specifies the year, followed by month, followed by day as yy_mm_dd, where an underscore (_) is the delimiter specified on the DATEDLM operand.

TIMEDLM
Specifies the character to be used as a delimeter to separate the hour, minutes, and seconds parts of the time where @@@@TIME is specified in the message text. The colon (:) is used if TIMEDLM is not specified. An ampersand (&) and single quotation mark (') are not valid delimiters.

USSEND macroinstruction

Use the USSEND macroinstruction to delimit the end of a USS table.

Syntax

```
►►──┬──────┬──USSEND──────────────────────────────────────────────►◄
    └─name─┘
```

Parameters

name
Specifies the name assigned to the macroinstruction.

Telnet INTERPRET table setup

This topic includes information about the Telnet INTERPRET table setup, including general rules and macroinstructions.

INTAB
The INTAB macroinstruction defines an interpret table that lists the Telnet application programs with which one or more logical units can establish a session. One INTAB macroinstruction defines the name of the interpret table and a group of logon messages definitions.

LOGCHAR
The LOGCHAR (logon-characters) macroinstruction defines a single logon message and the name of a host application program. More than one LOGCHAR can be included in an interpret table.

General usage rules for Telnet INTERPRET macroinstructions

Observe the following general usage rules for Telnet INTERPRET macroinstructions

- The Telnet interpret macroinstructions can be coded exactly as the VTAM macroinstructions. Telnet supports all functions supported by VTAM.
- An assembled and linked VTAM interpret table can be used directly by Telnet.
- For additional information about installing or changing an interpret table, see z/OS Communications Server: SNA Resource Definition Reference, which contains instructions for using the Telnet solicitor or USS Logon Panel.
- A sample interpret table is located in SEZAINST(EZBTPINT).
- The INTERPRET macros can be found in *hlq*.SISTMAC1, the VTAM macro library.

INTAB macroinstruction

Use the INTAB macroinstruction to define an interpret table that lists the VTAM application programs with which one or more logical units can establish a session. One INTAB macroinstruction defines the name of the interpret table and a group of logon message definitions.

Syntax

```
►►──┬──────┬──┬──────┬──INTAB─────────────────────────────────────►◄
    └─name─┘  └─NAME─┘
```

Parameters

name
Specifies an optional name for the macroinstruction. If specified, *name* must be unique and should be used as the operand for the assembler language END statement. When the macroinstruction is assembled, this name is used to identify the entry point to the interpret table CSECT.

LOGCHAR macroinstruction

Each LOGCHAR (logon-characters) macroinstruction defines a single logon message and the name of an application program, a logon interpret routing, or a USERVAR. You can include more than one LOGCHAR macroinstruction in an interpret table.

Syntax

```
►►──┬──────┬──LOGCHAR──APPLID──=──┬─(──APPLICID──,──application_name──)─┬──────────►
    └─name─┘                      ├─(──ROUTINE──,──routine_name──)──────┤
                                  └─(──USERVAR──,──uservar_name──)──────┘

   ┌─REMOVE──=──NO────────┐
►──┼──────────────────────┼──┬──────────────────────────────┬───────────────────────►◄
   └─REMOVE──=──┬─NO──┬───┘  └─SEQNCE──=──'──characters──'─┘
                └─YES─┘
```

Parameters

name
Specifies an optional name on the macroinstruction. This name is not used by Telnet and is ignored.

APPLID
Specifies the name of an application program, a logon interpret routine, or a USERVAR.

(APPLICID,*application_name***)**
Specifies the name of the application program. *application_name* can be any of the following values:

- ACBNAME of an application program in this host
- *applname* of an application program in this host
- *applname* of an application program in another host
- USERVAR representing an application program

application_name can be a network-qualified name. A network-qualified name takes the form of *netid.application_name*. If *application_name* is network-qualified, the network identifier is considered real and is not allowed to change. The resource name of the network-qualified name is considered Generic and can undergo USERVAR translation.

Restriction: If ACBNAME and the network name on the APPL definition statement for the application program are different, you cannot use a network-qualified ACBNAME.

(ROUTINE,*routine_name***)**
Specifies the routine name of the associated logon-interpret routine. All logon-interpret routines specified in an interpret table must be assembled and link-edited with that interpret table.

(USERVAR,*user_var_name***)**
The same as specifying APPLICID.

REMOVE=YES
Specifies that Telnet is to remove the first nonblank set of characters from the user logon sequence data being processed. The remaining data is left-aligned and padded with blanks on the right. You can substitute Y for YES when coding this parameter.

REMOVE=NO
Specifies that Telnet is not to remove any data from the user logon sequence. You can substitute N for NO when coding this parameter.

For example, if the following information is sent and REMOVE=Y is specified, Telnet removes “IMS10” before it passes the information to the application program in the user data field of the CINIT RU.

```
IMS10 NAME PASSWORD   =====>    NAME PASSWORD
```

SEQNCE
Specifies the required part of a logical unit's logon message.

The logon message might have additional data beyond the characters specified in the LOGCHAR macroinstruction. That data can be used and possibly changed by the logon-interpret routine if the ROUTINE operand is specified. Whether or not the data is changed or if a routine is called at all, the data is passed to the application program as user data.

To specify an apostrophe (') or an ampersand (&) within the logon message, code a double apostrophe ('') or a double ampersand (&&) within the character string. If the terminal user enters the logon message in lowercase and the message is not translated to uppercase (for example, by USS translate table), the value for '*characters*' must be coded in lowercase.

Do not specify leading and trailing device-control characters within a character string that is to be interpreted, because the USS facility deletes these characters. Device control characters coded within a logon message are deleted; therefore, a blank should not be coded for each occurrence of these characters. However, if a character within the logon message is translated to a blank by the interpret table, code a blank to represent that character.

LOGCHAR without SEQNCE or with SEQNCE='*' is considered a default match to the logon message. Telnet accepts the logon message and requests logon to the application program specified in the LOGCHAR macroinstruction. Therefore, place a default match LOGCHAR macroinstruction at the end of the interpret table. Otherwise, the remaining logon messages in the interpret table are not compared with the logon message entered by the terminal user.

Guideline: If you use two or more LOGCHAR macroinstructions, arrange them so that their SEQNCE fields are in reverse collating order.

Usage notes

- Telnet compares the logon message (character by character) with successive entries in the specified interpret table. If the leading characters in the logon message correspond to all the characters in an entry in the interpret table, Telnet accepts the logon message as valid (even though the logon message can be longer than the corresponding entry in the interpret table). If the first character or characters of several logon messages are identical, you should arrange the LOGCHAR macroinstructions so the logon sequences for the logon messages are from the most restrictive (greatest number of characters) to the least restrictive (fewest number of characters). For example:

```
SEQ1 LOGCHAR APPLID=(APPLICID,AP2),SEQNCE='LOG2'
SEQ2 LOGCHAR APPLID=(APPLICID,AP1),SEQNCE='LOG'
```

- Otherwise, in the preceding example, if sequence LOG had preceded LOG2 in the interpret table, both logon messages LOG and LOG2 would be valid logons to application program AP1. If you use two or more LOGCHAR macroinstructions, they must be arranged so that their SEQNCE fields are in reverse collating order.

Coding LOGON-INTERPRET routines

You can code logon-interpret routines to validate logons and determine the name of the application program that is to receive the logons. The entry point name must match the *routine name* specified in the APPLID=(ROUTINE,*routine name*) operand in the LOGCHAR macroinstruction. All logon-interpret routines specified in an interpret table must be assembled and link-edited with that interpret table.

The logon-interpret routine interface allows the routine to supply a network-qualified application name for interpreted logons.

If you want the logon-interpret routine to supply a network-qualified application name, you need to change the interpret routine parameter list. If you do not want the routine to supply a network-qualified name, you do not need to change the routine parameter list. You can use Registers 0 and 1 to supply the application name.

Requirements for logon-interpret routines

Entry from:
Telnet

Entry point:
routine name

Contents of registers at entry:

Register 0:
Length of logon message (any length from 1 to 80)

Register 1:
Address of first byte of logon message. For LOGON requests, Telnet searches the interpret table again, after USS translation, looking only for the specified APPLID. After USS translation, register 1 contains the address of the first byte of the APPLID.

Register 2:
Address of an 8-byte logical unit name

Register 4:
Address of parameter list for the network identifier and resource name.

Register 13:
Address of a 72-byte save area provided by Telnet.

Register 14:
Return address

Register 15:
Address of entry point of this routine.

Contents of Registers at Exit: If the interpreted name in the parameter list is blank, Registers 0 and 1 contain the name of the VTAM application program (in EBCDIC characters) with which Telnet is to establish a session:

Register 0:
First 4 characters of name (left-aligned).

Register 1:
Last 4 characters of name (left-aligned).

Registers 2–14:
Restored to condition at entry.

Register 15:
Return code:

00 Application program was found and the name is placed in registers 0 and 1.

Non0 Application program was not found and the name is not placed in registers 0 and 1.

If the name of the application program contains fewer than 8 characters, use blanks to provide a name with 8 characters.

Logon-interpret routine parameter list

When the exit gets control, the address of the following parameter list is in register 4. Offsets 0 through 27 include information about the fixed or interpreted name. Offset 28 includes the uninterpreted name.

Table 41. Logon interpret routine parameter list

Dec offset	Size (bytes)	Description	Input or output
0	2	Length of parameter list	Input
2	8	Name of requesting LU	Input
10	17	Interpreted name (in the form or either name or netid.name)	Output
27	1	Length of uninterpreted name	Input
28	n	Uninterpreted name	Input

Operation: The logon-interpret routine is run synchronously in pageable storage under the control of Telnet and not under the control of an application program. For the application program to receive the logon, this routine must validate the logon, obtain the name of the application program to receive control, and provide this name back to Telnet. Otherwise, the routine specifies that the logon is not valid or that the name of the application program was not found inTelnet.

The logon-interpret routine must also:

- Save and restore the contents of registers 2–14 when receiving and passing control.
- Use re-enterable code (the routine must not store anything within itself or modify itself during execution).

- Perform no I/O operations; an I/O request causes the routine to terminate abnormally.

The routine gets control in supervisor state with a Telnet storage key, so errors within the routine could cause damage to Telnet or to system control blocks and modules.

You can modify the logon message pointed to by register 1 that is passed to the interpret routine. However, remember these two points:

- Telnet does not look at the changed storage; it is passed as user data to the application.
- You should modify with caution, as modification outside the message storage boundaries could result in Telnet or TCP/IP stack outages.

The uninterrupted Logon message in the parameter list should not be changed as it is not passed as user data to the application.

ENDINTAB macroinstruction

Use the ENDINTAB macroinstruction to define the end of an interpret table. Code one ENDINTAB macroinstruction after one or more LOGCHAR macroinstructions to define the end of an interpret table. You can also follow the ENDINTAB macroinstruction with an assembler language END statement.

Syntax

```
►►──┬──────┬──┬──────┬──ENDINTAB─────────────────────────────────────►◄
    └─name─┘  └─NAME─┘
```

Parameters

name
: Specifies an optional name on the macroinstruction. This name is not used by Telnet and is ignored.

Usage notes

- If you code an assembler language END statement, it must be in the format:

  `END name`

  where *name* is the label of the INTAB macroinstruction and specifies the main entry point.
- Follow the ENDINTAB macroinstruction with an assembler language END statement unless the interpret table is to be followed by CSECTs containing one or more user-written APPLID routines.

Telnet LU exit setup

You can code LU exit routines to specify the LU name used to represent the client. You can optionally return a USS table name (3270 or SCS format) or an Interpret table name to be used by Telnet. The entry point name must match the routine name specified as the LUGROUP group name. Each LU exit routine specified must be assembled and link-edited as a stand-alone load module.

The LU exit can be driven multiple times. If the LUNAME returned by the exit cannot be registered, the exit is driven again for a different LUNAME. If the same name is returned, the connection fails. If a different name is returned, it is registered. If registration fails for the second name, the exit is driven a third time, and the *no additional retry flag* is activated. If the second name is again returned, the connection is terminated. If a third name is returned, registration is attempted again. If registration fails, the connection is terminated.

The LUNAME exit can be mapped as GENERIC or SPECIFIC. If both mappings are present, and the client presents Telnet with an LUNAME, the SPECIFIC mapping is attempted. If this attempt fails, the GENERIC mapping is also attempted. This conforms with the process used when TELNET assigns LUNAMEs.

Telnet LU exit setup operation

The LU exit routine runs synchronously in pageable storage under the control of Telnet; it is not under the control of the application program. The LU exit Routine can use non-reentrant code. Telnet ensures that only one process at a time calls the LU exit so it can maintain local storage in the routine for LU name management. The LU exit cannot perform I/O operations. An I/O request causes the routine to terminate abnormally. The routine gets control in supervisor state with the Telnet storage key. Errors in the LU exit might damage Telnet or the entire TCP/IP stack. Telnet monitors the number of abends by the LU exit. If 3 abends occur within a 10-minute period, the LU exit is disabled by Telnet. Telnet fails any future LU exit lookup without calling the LU exit.

Mapping rules apply to the LU exit as if it were an LU group. For example, if the LU exit is mapped to a Client Identifier as a Specific group, only connections requesting specific LUs use the LU exit. The only difference between an LU group and an LU exit is whether Telnet or the LU exit generates the LU name to use. At this time, the LU exit must be used on the LUMAP or PRTMAP statement alone. If Associated Printer function is being used on the LUMAP statement, neither the LU group nor the PRT group can be an LU exit. If the LU assigned to the connection is defined in LU groups mapped by both a LUG statement and an LUMAP/PRTMAP statement, neither LU group can be defined as an LU exit.

In addition to the several Client Identifiers passed to the LU exit using the parameter list pointed to by Register 1, the parameter list also includes any LU names or ranges that were coded in the LUGROUP and the requested application name, if specified. Telnet does not use the LU list. The LUGROUP can be defined without any LUs specified. The LUs specified can be used as seed values if the LU name exit wants to use them.

Version 2 of the LU exit function supports USS/Interpret table name specification. The parameter list for version 2 has been expanded to include specification of a 3270 format USS table name, SCS format USS table name, or an Interpret table name. Be sure your LU Exit checks the version number before accessing the expanded parameter list area. These fields are filled in with the mapped values, if they exist, by Telnet before the LU exit is called. If a name or names are changed upon return, Telnet attempts to load the table into storage if not already loaded.

Rules: The following rules apply:

- USS/Interpret table names are honored only for TN3270E connections. For TN3270E connections, the LU exit assigns an LU during connection negotiation. The LU exit is able to specify the USS table before end users receive their first USSMSG10 screen. For TN3270 connections or connections with the SIMCLIENTLU option defined, the LU exit does not assign an LU until an application is chosen. In these cases, the end user receives a Telnet solicitor panel or a USSMSG10 message from a profile-mapped USS table. When the LU exit is called for these connections, the Pl_UssIgnored flag is on, indicating that the USS tables or Interpret tables assigned by the LU exit are ignored.

- Telnet loads the USS/Interpret tables the first time they are assigned by the LU exit. If a table fails to load, the table mapped by the profile is used. If no profile mapping exists, a solicitor panel is sent to the client. If both 3270-format and SCS-format tables are specified by the exit, both tables must successfully load for the pair to be used.
- After the USS/Interpret tables that were assigned by the LU exit are loaded, Telnet replaces the currently assigned profile-mapped USS/Interpret tables (3270 or SCS format) with the LU exit tables. Telnet uses the new tables for all USS messages and commands.
- Setting the table name field to blanks indicates to Telnet that no table should be used for that table type. For example, if the profile mapping maps the EZBTPUST/EZBTPSCS value to a connection and the exit returns the EZBTPUST/, Telnet uses EZBTPUST only.
- The LU exit can assign USS/Interpret tables for TN3270E connections only. SIMCLIENTLU must not be coded. If an SCS-format table name is specified, that table is used; otherwise the 3270 format table is used.

The LU exit table specified remains in effect until the connection is dropped.

You can determine the USS tables used by a particular LU exit by issuing an OBJ display command for the LuGroup that is the LU exit. For example, if you defined the following LU group LUGROUP MyLuExit,EXIT ENDLUGROUP, you can issue a D TCPIP,,TELNET,OBJ,ID=MyLuExit command to view all USS tables loaded to support the LU Exit.

You can make changes to a USS table, and the changes become effective after the next V TCPIP,,OBEYFILE is issued. Whenever a V TCPIP,,OBEYFILE is issued, the USS tables specified by the LU Exit are reloaded.

Telnet specifies the function code in Register 0.

The following function codes are used:

- Function code 01 indicates the LU exit should create an LU name. Any algorithm can be used in the LU exit to generate an LU name. The LU exit either returns the LU name in the LU name field of the parameter list with a return code of 0 in Register 15, or the LU exit indicates that no LU name should be used and specifies a return code of 8 in Register 15. If Register 15 is 0, Telnet uses the LU name value, tries to register the LU name in the Telnet master LU database, and then assigns the LU to the connection. At this time, any nonzero return code is treated by Telnet as an indicator that the function did not work.

  Guideline: Use 8. In future releases, other values might be used to indicate specific reasons.
- Function code 02 indicates the LU name is no longer representing the connection and is being unassigned from the Telnet connection. The LU name is now available for assignment to another connection. It is up to the LU exit to manage the list of available LU names. If LU names are not reused, the LU exit might ignore the unassign function code. Whether or not the LU exit records the state change, Telnet ignores the return code value and deregisters the LU name from the Telnet master LU database.
- Function code 03 indicates the LU name is being deactivated because the operator issued the V TCPIP,,T,INACT,luname command or the ACB failed to

open. If the LU exit is tracking the state of LU names, an inactive LU should be considered not available to represent a client. Whether or not the LU exit changes the LU state within the exit, Telnet ignores the return code value, adds the LU name to the inactive LU list, and does not allow it to be registered in the master LU database.

- Function code 04 indicates the LU name is being activated because the operator issued the V TCPIP,,T,ACT,luname command. If the LU exit is tracking the state of LU names, the LU name should be considered available to represent a client. Whether or not the LU exit changes the LU state within the exit, Telnet ignores the return code value, removes the LU name from the inactive LU list, allows registration in the master LU database, and allows assignment to a Telnet connection.
- Function code 05 indicates the LU name is already in use. If the exit returns a different LU name, Telnet attempts to register this new LU name in the Telnet master LU name database. Telnet tries up to three LU names. If the third LU name is in use, Telnet notifies the exit with a flag bit indicating that no additional retries are attempted. Upon return from the third notification, Telnet does not look at the LU name field and fails the connection. Other connections have access to the exit between retry attempts.

If a specific LU name was requested by the client and it is not an LUGROUP name, that LU name is in the LU name field of the parameter list as input to the Exit. The LU exit can leave that LU name or override it with another name. In either case, Telnet then attempts to register the returned LU name.

If the LU name is already assigned or has been deactivated, Telnet fails the connection but does not notify the LU exit that the LU was already in use. If the OPEN ACB fails, Telnet notifies the Exit that the LU name is being deactivated in the Telnet Registration database by calling the LU exit with function code 03. If the LU name is activated using the Telnet ACT command, the LU exit is called with function code 04, indicating the LU name is reactivated.

Requirements for LU exit routines

This topic lists the requirements for LU exit routines.

```
Entry from:   Telnet
Entry point:  Routine name
```

Contents of registers at entry

The contents of registers at entry are as follows:

```
Register 0: Function code.  01 - Assign LU
                            02 - Unassign LU
                            03 - Inact LU
                            04 - Act LU
                            05 - LU in use
Register 1:  Address of parameter list specifying LU name, LUGROUP, and
             client known information.

Register 13: Address of a 72-byte save area provided by Telnet.

Register 14: Return address.

Register 15: Address of entry point of this routine.
```

Contents of registers at exit

The contents of registers at exit are as follows:

```
Registers 0-14: Restored to condition at entry.

Register 15:    Return code:
                00 - Use the LU name in the parameter area.
                08 - LU name is not to be used
```

If the name of the LU contains fewer than 8 characters, pad with blanks to the right to provide a name with 8 characters. The LU exit routine must save and restore the contents of registers 2-14 when receiving and passing control. Do not modify any values in the parameter list other than the LU name field. Do not alter more than the 8 bytes needed for the LU name. The R15 return code indicates to Telnet what action to take.

LU exit routine parameter list

When the exit gets control, the address of the following parameter list is in register 1:

```
Dec    Size   Description                      Input     Assign/Unassign   Inact/ACT
Offset (Bytes)                                   or      Inact (ACB fail)  Command
                                               Output

                                                         Value if          Value if
                                                          not set           not set

  0     8    LU name                          Both        Blanks            Always present
  8     1    Flag Bytes
               '80'x - 1 Client is a printer  Input       Always present    Always 0
                       0 Client is a terminal
               '40'x - 1 No additional retry  Input       Always present    Always 0
                       0 retry will be allowed
               '20'x - 1 IP Address is in IPv6 Input      Always present    Always 0
                         format
                       0 IP Address is in IPv4 format
               '10'x - 1  USS/SCS/Int tables   Input      Always present    Always 0
                       assigned by exit ignored for
                       this connection
                       0  USS/SCS/Int tables assigned
                       by exit will be used
  9     1    Parameter list Version level     Input       Always set        Always set
 10     1    Available byte                   Both        Always 0          Always 0
 11     1    Available byte                   Both        Always 0          Always 0
 12    16    Client IP address in hex         Input       Always set        Always 0
 28     4    Client Port                      Input       Always set        Always 0
 32    16    Destination IP address in hex    Input       Always set        Always 0
 48     4    Destination Port                 Input       Always set        Always 0
 52    16    Linkname                         Input       Blanks            Always Blanks
 68     8    Userid from Client Certificate   Input       Blanks            Always Blanks
 76     4    Ptr to hostname structure        Input       0                 Always 0
 80     8    Application netid                Input       Blanks            Always Blanks
 88     8    Application name                 Input       Blanks            Always Blanks
 96     8    Userid from solicitor panel      Input       Blanks            Always Blanks
104     4    Ptr to LUGroup structure         Input       0                 0
108     8    USS table name - 3270 format     Both        Blanks            Always Blanks
116     8    USS table name - SCS format      Both        Blanks            Always Blanks
124     8    Interpret table name             Both        Blanks            Always Blanks
132    16    Reserved                         Both        Blanks            Always Blanks

Hostname structure

  0     1    Total length of Hostname         Input       0
  1     255  Client hostname                  Input

LuGroup  structure

  0     4    Number of single LU names        Input
  4     n    List of all single LU names,
             each 8 characters
             (n=8*number of LUs)             Input       0
 n+4    4    Number of LU range structures    Input       0
 n+8    m    List of all LU range structures,
             each 24 characters (low/high/variant)
             (m=24*number of structures)       Input
```

Chapter 17. EXPRESS LOGON using DCAS

The Digital Certificate Access Server (DCAS) is a host-based server that provides some distributed z/OS security server services. The most common service is Pass Ticket (like a password) generation services. It typically works in conjunction with SSL-authenticated clients that provide logon services on behalf of end users (typically workstation users) that want to log on to host applications. This allows users to log on to host applications without having to know their password, and possibly even their user ID. On the host, DCAS works with the resident security server, such as RACF, to provide this function.

Tip: You can customize the DCAS to use IBM System SSL, but it is recommended to implement TLS security by using AT-TLS policies. For more information about using AT-TLS policies, see Customizing DCAS for TLS/SSL in z/OS Communications Server: IP Configuration Guide.

Requirement: The application Pass Ticket generation must be configured in RACF.

DCAS can support several different client-types for express logon. For additional overview and configuration information about Express Logon, see z/OS Communications Server: IP Configuration Guide.

This topic contains the following information:
- "Starting Digital Certificate Access Server"
- "Digital Certificate Access Server (DCAS) environment variables" on page 687
- "Digital Certificate Access Server (DCAS) sample procedure (EZADCASP)" on page 686
- "PassTicket server configuration file processing when using IBM System SSL" on page 687
- "Digital Certificate Access Server (DCAS) configuration file keywords and parameters" on page 688

Starting Digital Certificate Access Server

You can start the DCAS from the z/OS UNIX shell or with an MVS started procedure using optional parameters for debugging, logging, and specifying the configuration file. To start the DCAS from the z/OS UNIX shell, use the following format:

```
dcas <parameter_1> <parameter_2> <parameter_3> &
```

To start the DCAS from an MVS started procedure, use the following format:

```
PARM=.../<parameter_1> <parameter_2> <parameter_3>
```

The following optional parameters can be used with both the DCAS UNIX command and the MVS started procedure:

`-d or -D`
Indicates debugging. The following levels apply:

1	Specifies log error and warning messages.
2	Specifies log error, warning, and informational messages.

3 Specifies log error, warning, informational, and debug messages. This is the default.

-l or -L
Indicates logging to SYSLOGD or to a designated log file. If you do not specify this parameter, logging defaults to /tmp/dcas.log.

If you specify a debug level, but not logging, then the DCAS attempts to open the default log file /tmp/dcas.log. If this fails, debugging is turned off.

For SYSLOGD, the DCAS uses the log facility local0.

-c or -C
Indicates the requested configuration file (for example, /u/userx/passtick.conf). If you do not specify this parameter, the DCAS looks for the configuration file using the following search order:

1. DCAS_CONFIG_FILE environment variable
2. /etc/dcas.conf
3. tsouserid.DCAS.CONF
4. TCPIP.DCAS.CONF

Restriction: If the DCAS does not find a valid configuration file, it does not start.

The /tmp/dcas.*tcpname*.pid is a temporary DCAS pid file that the DCAS creates. This file contains the process ID of the current invocation of the DCAS.

Restrictions:

- If /tmp/dcas.*tcpname*.pid is a symbolic link, it must have an owning UID or GID that matches the EUID or EGID that is assigned to the DCAS.
- If /tmp/dcas.*tcpname*.pid is a hard link or the target of a hard link, users that are outside the owner or group of the directory in which /tmp/dcas.*tcpname*.pid is stored cannot have write access to the directory. Additionally, write access to /tmp/dcas.*tcpname*.pid must be limited to the owning UID or group, for example, --w--w----permissions.

Digital Certificate Access Server (DCAS) sample procedure (EZADCASP)

```
//DCAS      PROC
//*
//*   IBM Communications Server for z/OS
//*   SMP/E distribution name: EZADCASP
//*
//*   5694-A01 (C) Copyright IBM Corp. 2000, 2005
//*   Licensed Materials - Property of IBM
//*   "Restricted Materials of IBM"
//*   Status = CSV1R7
//*
//*   Function: Sample procedure for running the Digital
//*             Certificate Access Server (DCAS)
//*
//DCAS   EXEC PGM=EZADCDMN,REGION=4096K,TIME=NOLIMIT,
//   PARM='POSIX(ON) ALL31(ON) / -d 1 -l SYSLOGD'
//*
//*** Notes:
//*
//* - DCAS can also be invoked from the Unix System Services shell
//*   as a shell command: dcas
//*
//* - The z/OS Secure Socket Layer (SSL) product libraries must
//*   be accessible at runtime to DCAS-  hlq.mlq.SGSKLOAD.
//*
//* - The system link list concatenation must contain the TCP/IP
//*   runtime libraries and the C runtime libraries.  If they are
//*   not in the link list concatenation, this procedure will need
//*   to be changed to STEPLIB to them.
//*
//* - To pass parameters to DCAS, specify them after the final slash
//*   on the PARM statement.  For example:
//*      //   PARM=('POSIX(ON) ALL31(ON)',
//*      //   'ENVAR("LIBPATH=/usr/lib")/-d 3 -l SYSLOGD')
//*
//* - Other examples
//* //     PARM=('POSIX(ON) ALL31(ON) TERMTHDACT(UATRACE) TRAP(ON)',
//* //     'ENVAR("DCAS_CONFIG_FILE=/u/us1/xxx.conf")/ -d 3 -l SYSLOGD')
//*
//*
//*
//STDENV   DD DUMMY
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Figure 27. DCA server configuration file sample

Digital Certificate Access Server (DCAS) environment variables

Table 42 provides a list of environment variables used by DCAS that can be tailored to a particular installation:

Table 42. DCAS environment variables

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules (or a link to syntax)
DCAS_CONFIG_FILE	DCAS (Digital Certificate Access Server)	Identifies the location of the DCAS configuration file	None

| PassTicket server configuration file processing when using IBM | System SSL

The following list shows some rules for processing a PassTicket server configuration file:

- The # symbol as the first character in a configuration file indicates a comment. The format for specifying keywords and values is <keyword> <value> with a space between the keyword and the value.
- If a keyword is unrecognized as one of the valid DCAS keywords, a message is sent to the console indicating a keyword that is not valid was detected, but that

processing continues. If the same keyword is specified more than one time in the profile, the last value specified for that keyword is used.

- You can use upper or lowercase for keywords and most values, but values representing file names in the z/OS UNIX are case sensitive.
- z/OS UNIX file names can be 128 characters or fewer in length. This includes the path name, file name, forward slashes (/), and periods (.). File names longer than 128 characters are truncated.

Digital Certificate Access Server (DCAS) configuration file keywords and parameters

This topic describes the keywords and parameters that are used in the DCAS configuration file. It also describes tips for each DCAS parameter for AT-TLS configuration when TLSMECHANMISM is set to AT-TLS. For more information about AT-TLS policies, see Required AT-TLS policies when DCAS TLSMECHANISM is set to ATTLS and Migrating existing DCAS server to use AT-TLS policies in z/OS Communications Server: IP Configuration Guide.

CLIENTAUTH

Use the CLIENTAUTH keyword and parameters to specify client authentication.

Tip: Use this keyword regardless of the value that is configured on the TLSMECHANISM keyword. For TLSMECHANISM ATTLS, you must also configure the following values in the corresponding AT-TLS policy:

- TTLSEnvironmentAction HandshakeRole ServerWithClientAuth
- TTLSEnvironmentAction -> TTLSEnvironmentAdvancedParms
  - LOCAL1 - ClientAuthType Required
  - LOCAL2 - ClientAuthType SAFCHECK

```
                   ┌─LOCAL2────────────┐
►►──CLIENTAUTH─────┴─LOCAL1──LOCAL2────┴──────────────────────────────────────►◄
```

Parameters

LOCAL1
Specifies that the SSL handshake process authenticates the client certificate as well as the server certificate. This check verifies the client has received a certificate from a trusted certificate CA.

LOCAL2
Specifies that the SSL handshake process authenticates the client certificate and provides additional access control through the installation's SAF-compliant security product (for example, RACF). The following conditions apply:

- LOCAL2 verifies the client certificate has an associated user ID defined to the security product. The certificate must first be defined to the security product to obtain this validation. For more information about adding certificates to RACF, see the description of the RACDCERT command in the z/OS Security Server RACF Command Language Reference.
- For security products that support the SERVAUTH class, installations can also obtain a more granular level of access control. If the installation has activated the SERVAUTH class and provided a profile for the DCAS in the

SERVAUTH class, only users specified in the profile are allowed to connect to the port. The security product profile name is specified using the following format:

```
EZA.DCAS.sysname
```

where *sysname* is the name of the MVS system image.

Tip: Client certificate refers to the DCAS Client:
- TN3270 middle-tier server in the case of the IBM Express Logon Feature (ELF)
- Host on Demand (HoD) or HATS for WebExpress Logon
- The client connecting to DCAS for other enhanced logon solutions

IPADDR

Use this keyword to define the IP address to which the DCAS binds.

Tip: Use this keyword regardless of the value that is configured on the TLSMECHANISM keyword. For TLSMECHANISM ATTLS, you must set the TTLSRule LocalAddr parameter in the corresponding AT-TLS policy to match the IPADDR value.

```
►►──┬────────┬──┬────────┬──────────────────────────────────────────►◄
    └─IPADDR─┘  └─ipaddr─┘
```

Parameters

ipaddr
Specifies the IP address or host name to which the DCAS binds. If you do not specify ipaddr, the DCAS binds to the IPv4 INADDR_ANY address or to the IPv6 unspecified address (in6addr_any).

Restriction: Scope information cannot be specified with the IP address or the host name.

Requirement: The TCP/IP stack must be IPv6-enabled if an IPv6 address (or host name which resolves to an IPv6 address) is specified.

KEYRING

Use the KEYRING keyword to define the z/OS UNIX file containing the certificate to be used during the SSL handshake.

Tip: Do not use this keyword if TLSMECHANISM is ATTLS. For TLSMECHANISM ATTLS, you must set TTLSEnvironmentAction -> TTLSKeyringParms Keyring in the corresponding AT-TLS policy with the value that was used on the KEYRING or SAFKEYRING keyword.

```
►►──KEYRING──hfsfilename────────────────────────────────────────────►◄
```

Parameters

hsffilename
Specifies the path and file name of the key ring file.

Usage notes

The keywords KEYRING and SAFKEYRING are mutually exclusive. If neither KEYRING nor SAFKEYRING is specified, the default key ring file, key.kdb, is used.

LDAPPORT

Use the LDAPPORT keyword to allow authentication of the client certificate by an X.500 host. LDAPPORT is used in combination with LDAPSERVER.

Tip: Do not use this keyword if TLSMECHANISM is ATTLS. For TLSMECHANISM ATTLS, set TTLSEnvironmentAction -> TTLSGskAdvancedParms -> TTLSGskLdapParms GSK_LDAP_SERVER_PORT in the corresponding AT-TLS policy to the LDAP port value.

```
►►──┬──────────┬──┬──────┬─────────────────────────────────────────────►◄
    └─LDAPPORT─┘  └─port─┘
```

Parameters

port
Specifies the port number of the X.500 host.

LDAPSERVER

Use the LDAPSERVER keyword to allow authentication of the client certificate by an X.500 host. LDAPSERVER is used in combination with LDAPPORT.

Tip: Do not use this keyword if TLSMECHANISM is ATTLS. For TLSMECHANISM ATTLS, set TTLSEnvironmentAction -> TTLSGskAdvancedParms -> TTLSGskLdapParms GSK_LDAP_SERVER in the corresponding AT-TLS policy to the correct LDAP server location.

Restriction: An IPv6 address is not supported for this keyword.

```
►►──LDAPSERVER──┬────────┬──fqname──────────────────────────────────────►◄
                └─ipaddr─┘
```

Parameters

fqname
Specifies the fully qualified host name of the X.500 host.

ipaddr
Specifies the dotted-decimal IP address of the X.500 host.

PORT

Use the PORT keyword to define a basic port to the DCAS.

Tip: Use this keyword regardless of the value that is configured on the TLSMECHANISM keyword. For TLSMECHANISM ATTLS, you must set the TTLSRule LocalPortRange parameter in the corresponding AT-TLS policy to match the PORT value.

```
►►──┬──────┬──┬──────────┬──────────────────────────────────────────────►◄
    └─PORT─┘  └─port num─┘
```

Parameters

8990
Specifies the port over which the DCAS accepts incoming requests. Port 8990 is the default.

num
Specifies a particular port number.

SAFKEYRING

Use the SAFKEYRING keyword to define the RACF-defined key ring containing the certificate to be used during the SSL handshake.

Tip: Do not use this keyword if TLSMECHANISM is ATTLS. For TLSMECHANISM ATTLS, you must set TTLSEnvironmentAction -> TTLSKeyringParms Keyring in the corresponding AT-TLS policy with the value of the keyring name.

```
►►──SAFKEYRING──keyringname─────────────────────────────────────────────►◄
```

Parameters

keyringname
Specifies the name to use when creating a key ring with the RACF ADDRING function. This name is case sensitive.

Usage notes

The keywords SAFKEYRING and KEYRING are mutually exclusive. If neither SAFKEYRING nor KEYRING is specified, the default key ring file, key.kdb, is used.

SERVERTYPE

Use the SERVERTYPE keyword and parameter to specify the type of input the DCAS server receives.

The SERVERTYPE keyword definition can be specified multiple times in the DCAS configuration file.

Subsequent definitions are logically ORed. For example, defining SERVERTYPE ALLTYPES and then SERVERTYPE NOUSERIDTYPE means that DCAS no longer accepts a user ID as input.

Tip: Use this keyword regardless of the value that is configured on the TLSMECHANISM keyword.

Restriction: Because SERVERTYPE CERTTYPE is the default, it is not valid to only specify SERVERTYPE NOCERTTYPE.

```
   ┌─SERVERTYPE CERTTYPE─┐
►►─┴─────────────────────┴─SERVERTYPE CERTTYPE──┬─ALLTYPES───────┬──────────►◄
                                                ├─USERIDTYPE─────┤
                                                ├─KERBEROSTYPE───┤
                                                ├─NOCERTTYPE─────┤
                                                ├─NOUSERIDTYPE───┤
                                                └─NOKERBEROSTYPE─┘
```

Parameters

Guideline: For any SERVERTYPE parameter, DCAS returns a Pass Ticket for the application name that it receives from the client.

SERVERTYPE CERTTYPE
Specifies that DCAS accepts only a X.509 certificate and application name as input. This is the default.

SERVERTYPE ALLTYPES
Specifies that DCAS accepts any form of currently supported and future inputs. This enables DCAS to accept a x.509 certificate, and application name (SERVERTYPE CERTTYPE) as well as a user ID and application name (SERVERTYPE USERIDTYPE).

SERVERTYPE USERIDTYPE
Specifies that DCAS accepts only a user ID and application name as input.

SERVERTYPE KERBEROSTYPE
Specifies that DCAS accepts a Kerberos principal name and application name as input.

SERVERTYPE NOCERTTYPE
Specifies that DCAS not accept the x.509 certificate and application name as input. You can use this to turn off a previous SERVERTYPE CERTTYPE parameter.

SERVERTYPE NOUSERIDTYPE
Specifies that DCAS not accept user ID and application name as input. You can use this to turn off a previous SERVERTYPE USERIDTYPE parameter.

Requirements: You must specify certain values for following IBM-enhanced logon solutions:

- Express Logon Feature (ELF) requires a SERVERTYPE value of CERTTYPE or ALLTYPES.
- Web Express Logon (WEL) requires a SERVERTYPE value of USERIDTYPE or ALLTYPES.

For enhanced logon solutions other than those listed, see your product documentation for the SERVERTYPE value you need to specify. You should have an understanding of the DCAS function required by the solution prior to configuring the SERVERTYPE parameter because the data that DCAS provides is highly sensitive.

SERVERTYPE NOKERBEROSTYPE
Specifies that DCAS does not accept a Kerberos principal name and application name as input. Use this parameter to turn off a previous SERVERTYPE KERBEROSTYPE parameter.

Usage notes

- The keywords SAFKEYRING and KEYRING are mutually exclusive. If neither SAFKEYRING nor KEYRING is specified, the default key ring file, key.kdb, is used.
- The KERBEROSTYPE support enables the DCAS client to provide a Kerberos principle name and application ID. The Kerberos principal name must be mapped to a RACF user ID. This allows DCAS to provide a Pass Ticket for the user ID and application name. See z/OS Security Server RACF Security Administrator's Guide for information about defining a KERBLINK profile.

  The DCAS server has been enhanced to provide the new function. This requires that the administrator of the single-signon solution use RACF or a similar security product to map a valid z/OS user ID to a Kerberos principal name. In RACF, do this by creating a KERBLINK profile in RACF.

  See z/OS Security Server RACF Security Administrator's Guide for a description of Kerberos principal names and how to map them to user IDs.

STASHFILE

Use the STASHFILE keyword to specify the key ring password file to the associated key ring file. This password file contains the encrypted password.

Tip: Do not use this keyword if TLSMECHANISM is ATTLS. For TLSMECHANISM ATTLS, if STASHFILE is required, you must set TTLSEnvironmentAction -> TTLSKeyringParms KeyringStashFile in the corresponding AT-TLS policy with the pathname of the stash file.

```
►►──STASHFILE──hfsfilename──────────────────────────────────────────────►◄
```

Parameters

hfsfilename
Specifies the path and file name of the password file.

Usage notes

STASHFILE is normally associated with the file used on the KEYRING parameter. The file name defaults to key.sth. The file is not needed for SAFKEYRING.

TCPIP

Use the TCPIP keyword to specify the active TCP/IP stack name with which the DCAS establishes affinity.

Tip: Use this keyword regardless of the value that is configured on TLSMECHANISM keyword.

```
►►──TCPIP──stackname────────────────────────────────────────────────────►◄
```

Parameters

stackname
Specifies the name of the TCP/IP stack with which the DCAS establishes affinity.

TLSMECHANISM

Use the TLSMECHANISM keyword and parameters to specify whether to use IBM System SSL directly or AT-TLS policies. It is recommended to use AT-TLS to implement TLS security. For more information about using AT-TLS policies, see Customizing DCAS for TLS/SSL in z/OS Communications Server: IP Configuration Guide.

```
                      ┌─DCAS──┐
►►──TLSMECHANISM──────┤       ├──────────────────────────────────────────────►◄
                      └─ATTLS─┘
```

Parameters

DCAS
IBM System SSL is used directly for TLS/SSL. This is the default.

ATTLS
AT-TLS policies are used for TLS/SSL.

Tip: For more information about AT-TLS policies, see Required AT-TLS policies when DCAS TLSMECHANISM is set to ATTLS and Migrating existing DCAS server to use AT-TLS policies in z/OS Communications Server: IP Configuration Guide.

TLSV1ONLY

Use the TLSV1ONLY keyword to control whether TLSv1.0 is the only supported SSL version for connections that are secured using SSL implemented by DCAS.

```
                   ┌─TRUE──┐
►►──TLSV1ONLY──────┤       ├─────────────────────────────────────────────────►◄
                   └─FALSE─┘
```

Parameters

TRUE
Indicates that only TLSv1.0 can be enabled. This is the default.

FALSE
Indicates that SSLV2, SSLV3, and TLSV1.0 can be enabled.

Rule

TLSV1ONLY is honored only when TLSMECHANISM DCAS is specified.

V3CIPHER

Use the V3CIPHER keyword to specify a subset of the supported SSL V3 cipher algorithms.

Tip: Do not use this keyword if TLSMECHANISM is ATTLS. For TLSMECHANISM ATTLS, if V3CIPHER is required, you must set TTLSEnvironmentAction -> TTLSCipherParms -> V3CipherSuites to the desired cipher values in the corresponding AT-TLS policy. For more information about cipher values, see Migrating existing DCAS server to use AT-TLS policies for V3Cipher in z/OS Communications Server: IP Configuration Guide.

```
►►──┬──────────┬──┬────────────┬──────────────────────────────────────────►◄
    └─V3CIPHER─┘  └─cipherspec─┘
```

Parameters

cipherspec

Specifies the level of the SSL V3 cipher to use for a DCAS (for example, V3CIPHER 0306090201). The following cipher levels are valid:

- 01 = NULL MD5
- 02 = NULL SHA
- 03 = RC4 MD5 Export
- 04 = RC4 MD5 US
- 05 = RC4 SHA US
- 06 = RC2 MD5 Export
- 09 = DES SHA
- 0A = Triple DES SHA US

Usage notes

If you do not specify V3CIPHER, it defaults to the cipher level supported by the SSL library installed on your system.

Steps for setting up RACF for Digital Certificate Access Server (DCAS)

This topic describes how to set up RACF for DCAS.

Procedure

Perform the following steps to set up RACF for DCAS:

1. Define a user ID as superuser to OMVS services.

   The server requires that you define the user ID from which the server is started to be defined to use OMVS services as a superuser. You can configure the OMVS(UID(0)) on the ADDUSER command. However, if the user ID already exists, the ADDUSER fails and the user ID is not altered to superuser. The ALTUSER value sets the user ID to superuser whether the user ID existed before or the ID was just created by the ADDUSER command.

   ```
   ADDUSER DCAS ALTUSER DCAS DFLTGRP(OMVS) OMVS(UID(0)HOME('/'))
   ```

2. Give the user ID access to operator commands.

   If the OPERCMDS class profile MVS.SERVMGR.DCAS is defined to control who can start DCAS, then the user ID that starts DCAS must have CONTROL access to the profile. Use the following commands to provide access:

   ```
   RDEFINE OPERCMDS(MVS.SERVMGR.DCAS) UACC(NONE)
   PERMIT MVS.SERVMGR.DCAS CLASS(OPERCMDS) ACCESS(CONTROL)
   ID(DCAS)
   SETROPTS RACLIST(OPERCMDS) REFRESH
   ```

3. Provide a RACF definition for MVS startup.

   If the server is started as an MVS procedure, use the following RACF definitions to define the server to RACF:

   ```
   RDEFINE STARTED DCAS.* STDATA(USER(DCAS)) SETROPTS RACLIST(STARTED) REFRESH
   ```

Chapter 18. File Transfer Protocol

This topic contains z/OS File Transfer Protocol (FTP) client and server configuration information and includes the following information:

- “FTP server cataloged procedure (FTPD)”
- “FTP server cataloged procedure (FTPD) parameters” on page 699
- “FTP configuration statements in FTP.DATA” on page 721
- “SOCKS configuration statements in SOCKSCONFIGFILE” on page 898

FTP clients and servers both use a configuration file, referred to as FTP.DATA. FTP.DATA can be used to customize FTP behavior. The server's FTP.DATA file customizes the behavior of the server system, and the client's FTP.DATA file customizes the behavior of the client system.

For example, if you want to create data sets on the server's system with a logical record length of 80 characters, and create data sets on the client's system with a logical record length of 256, perform the following steps:

1. Specify the LRECL 80 configuration statement in the FTP server's FTP.DATA configuration file.
2. Specify the LRECL 256 configuration statement in the FTP client's FTP.DATA configuration file.

The setting in the FTP server's configuration file is used by the server when the user creates a file on the server's system with an FTP subcommand, such as PUT. Likewise, the setting in the FTP client's configuration file is used by the FTP client when the user creates a file on the client's system with an FTP subcommand, such as GET.

Guideline: The client setting does not override the server setting. Instead, the server setting affects data sets created on the server's system, while the client setting affects data sets created on the client system.

FTP server cataloged procedure (FTPD)

The following sample shows a start procedure for the FTP server. No start procedure is required for the FTP client.

```
//FTPD   PROC MODULE='FTPD',PARMS=''
//*********************************************************************
//*        Descriptive Name:           FTP Server Start Procedure   *
//*        File Name:                  tcpip.SEZAINST(EZAFTPAP)     *
//*                                    tcpip.SEZAINST(FTPD)         *
//*        SMP/E Distribution Name:    EZAFTPAP                     *
//*                                                                 *
//*        Licensed Materials - Property of IBM                     *
//*        "Restricted Materials of IBM"                            *
//*        5694-A01                                                 *
//*        (C) Copyright IBM Corp. 1995, 2005                       *
//*        Status = CSV1R7                                          *
//*********************************************************************
//*
//* SET PARM1=TCPIVP.TCPPARMS(TCPDATA)
//*
//FTPD   EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//       PARM='/&PARMS'
//*
//*   Uncomment the SET statement above when using the next two lines.
//*   PARM=('ENVAR("RESOLVER_CONFIG=//''&PARM1''")',
//*   '/&PARMS')
//*
//*   PARM=('ENVAR("_BPX_JOBNAME=myftp")',
//*   '/&PARMS')
//*
//*   PARM=('ENVAR("KRB5_SERVER_KEYTAB=1")',
//*   '/&PARMS')
//*
//**** IVP Note *****************************************************
//*
//* If executing the FTP installation verification procedures (IVP),
//* - Comment the first PARM card and uncomment both lines of the
//*   second PARM card
//* - Uncomment the appropriate SYSFTPD and SYSTCPD DD cards for the IVP
//*
//*******************************************************************
//**** _BPX_JOBNAME Note ********************************************
//*
//* The environment variable _BPX_JOBNAME can be specified
//* here in the FTPD procedure, so that all of the logged on
//* FTP users will have the same jobname.  This can then
//* be used for performance control and identifying all FTP users.
//* To use this:
//* - Comment the first PARM card and uncomment both lines of the
//*   third PARM card
//*
//*******************************************************************
//**** KRB5_SERVER_KEYTAB Note **************************************
//*
//* The environment variable KRB5_SERVER_KEYTAB can be specified
//* here in the FTPD procedure, so that the FTP server will use the
//* local instance of the Kerberos security server to decrypt tickets
//* instead of obtaining the key from the key table.
//* To use this:
//* - Comment the first PARM card and uncomment both lines of the
//*   fourth PARM card
//*
//*******************************************************************
//*
//*       The C runtime libraries should be in the system's link
//*       list or add them to the STEPLIB definition here.  If you
//*       add them to STEPLIB, they must be APF authorized.
//*
//*       To submit SQL queries to DB2 through FTP, the DB2 load
//*       library with the suffix DSNLOAD should be in the system's
//*       link list, or added to the STEPLIB definition here.  If
//*       you add it to STEPLIB, it must be APF authorized.
//*
//CEEDUMP  DD SYSOUT=*
//*
//*       SYSFTPD is used to specify the FTP.DATA file for the FTP
//*       server.  The file can be any sequential data set, member
//*       of a partitioned data set (PDS), or HFS file.
//*
//*       The SYSFTPD DD statement is optional.  The search order for
//*       FTP.DATA is:
//*
//*           SYSFTPD DD statement
//*           jobname.FTP.DATA
//*           /etc/ftp.data
//*           SYS1.TCPPARMS(FTPDATA)
//*           tcpip.FTP.DATA
//*
//*       If no FTP.DATA file is found, FTP default values are used.
//*       For information on FTP defaults, see z/OS Communications
//*       Server: IP Configuration Reference.
//*SYSFTPD DD DISP=SHR,
//*        DSN=TCPIP.SEZAINST(FTPSDATA)
//*SYSFTPD DD DISP=SHR,
//*        DSN=TCPIVP.TCPPARMS(FTPSDATA)
//*
//*       SYSTCPD explicitly identifies which data set is to be
//*       used to obtain the parameters defined by TCPIP.DATA
//*       when no GLOBALTCPIPDATA statement is configured.
//*       See the IP Configuration Guide for information on
//*       the TCPIP.DATA search order.
//*       The data set can be any sequential data set or a member of
//*       a partitioned data set (PDS).
//*SYSTCPD DD DISP=SHR,
//*        DSN=TCPIP.SEZAINST(TCPDATA)
//*SYSTCPD DD DISP=SHR,
//*        DSN=TCPIVP.TCPPARMS(TCPDATA)
//*
```

Figure 28. Sample start procedure for the daemon

FTP server cataloged procedure (FTPD) parameters

The system parameters required by the FTP server are passed by the PARM parameter on the EXEC statement of the FTPD cataloged procedure. Add your parameters to `PARMS='` in the PROC statement of the FTPD cataloged procedure, making certain that:

- Each parameter is separated by a blank.
- All parameters are in uppercase, unless the security product administrator has enabled mixed-case password support. In that case, any password you supply as a parameter must be entered in the correct case.

For example: `//FTPD PROC MODULE='FTPD',PARMS='TRACE ANONYMOUS PORT 21'`

`ANONYMOUS`
Specifying this start option is equivalent to coding the ANONYMOUS statement in FTP.DATA with no parameters. See "ANONYMOUSLEVEL (FTP server) statement" on page 748 and "ANONYMOUS (FTP server) statement" on page 738 for more information.

`ANONYMOUS=``*user_id*`
Specifying this start option is equivalent to coding the ANONYMOUS *user_id* statement in FTP.DATA. See "ANONYMOUSLEVEL (FTP server) statement" on page 748 and "ANONYMOUS (FTP server) statement" on page 738 for more information.

`ANONYMOUS=``*user_id/password*`
Specifying this start option is equivalent to coding the ANONYMOUS *user_id/password* statement in FTP.DATA. See "ANONYMOUSLEVEL (FTP server) statement" on page 748 and "ANONYMOUS (FTP server) statement" on page 738 for more information.

Restriction: Do not code a password phrase as *password*.

`ANONYMOUS=``*user_id/SURROGATE*`
Specifying this start option is equivalent to coding the ANONYMOUS *user_id/SURROGATE* statement in FTP.DATA. See "ANONYMOUSLEVEL (FTP server) statement" on page 748 and "ANONYMOUS (FTP server) statement" on page 738 for more information.

Requirement: To use this option, ANONYMOUSLEVEL must be greater than or equal to 3.

`AUTOMOUNT`
Permits a DASD volume to be mounted when attempts are made to access data sets on that volume.

`AUTORECALL`
Permits data sets migrated by a storage manager, such as hierarchical storage manager (HSM), to be recalled automatically.

DATASETMODE
Treats all lower qualifiers of address space names as part of the same directory. This affects the behavior of DIR, LS, MGET, and MDLETE because all lower qualifiers are returned.

DIRECTORYMODE
Treats each level of an address space name as if it were a directory. This affects the behavior of DIR, LS, MGET, and MDLETE because only the next lower qualifier is returned.

INACTIVE *number_seconds*
Sets the inactivity timeout to the specified number of seconds. A control connection inactive for this amount of time is closed. The default inactivity timeout is 300 seconds (5 minutes). The maximum inactive time is 86 400 seconds. A value of 0 disables the inactivity timer, and inactive control connections do not time out.

NOAUTOMOUNT
Prevents a DASD volume from being mounted when attempts are made to access data sets on that volume.

NOAUTORECALL
Prevents data sets migrated by a storage manager, such as HSM, from being recalled automatically. Migrated data sets can still be deleted even though NOAUTORECALL is specified.

Restriction: Only sequential and whole partitioned data sets can be deleted without recalling. Partitioned data set members require the whole data set to be recalled.

PORT *port_num*
Accepts incoming requests on the specified (decimal) port number rather than the port specified in /etc/services or the default port of 21. (*port_num* − 1) is used for data transfer. The maximum port number is 65534.

TRACE
Running TRACE might affect performance and should only be used when diagnosing problems with FTP sessions.

FTP server user exits

To limit access to an FTP server, you can use any of the user exits described in this topic. The FTP server provides increased security by using user exits.

A user exit is passed the address of a parameter list in register 1. The parameter list is a series of pointers to values. The first word of the parameter list always points to the return code. If the user exit sets the return code to 0, processing continues as normal. If the return code is not 0, authorization is denied and the user receives a negative reply indicating that the command has failed. Upon entry, the return code is 0, so a correct return can be indicated by leaving the return code alone. The return code field in the FTPOSTPR exit is included for consistency; it has no effect on processing.

The second word of the parameter list always points to a word containing the number of parameters that follow. This helps handle any future releases that might increase the number of parameters in these parameter lists.

The remainder of the parameter list points to values the FTP user exit uses in its processing.

Requirements:

- The user exit load modules must be in a cataloged data set and placed in an APF-authorized library to which the FTP server has access by way of STEPLIB, linklist, or LPA.
- The authorization state (JSCBAUTH) must be the same after exiting from the user exit as it was upon entry.
- User exit routines must be reentrant.
- User exit routines are invoked in TCB mode, problem program state, with AMODE(31). If the user exit routine changes a setting, the user exit routine must restore the setting before returning to the caller.
- The FTPCHKIP user exit is loaded when the FTP daemon initializes. If you want the FTP daemon to use a new version of this exit routine, you must stop the FTP daemon and start it again.

Rule: All data areas that are passed to the exit, including the Language Environment save area stack, above the 31 bit addressing line. If the exit routine uses any system services that require data areas below the 24 bit addressing line, the exit routine must obtain the necessary storage below the line and copy any data values there.

Guidelines:

- If you are debugging a user exit routine, you should have a test version of a server to work with so that you can stop and start without affecting other users. You can do that by putting a PORT parameter in the EXEC statement of the FTP JCL, such as PARMS='PORT 1073'. To connect to this server, enter the following code:

  ```
  FTP remoteHost 1073
  ```

  You can use any number as a port number for your test FTP server. IBM suggests that you choose a number that does not conflict with any well-known port numbers used on your host.
- z/OS FTP follows the MVS search order to load the FTP exit routines. If you are not using the user exit facility, put a dummy user exit load module in the first library in the MVS search order. This prevents other users from putting their own modules in a library later in the concatenation sequence. This also increases the need to have that library protected using SAF.

Restriction: You cannot use the System Programming C Facilities for the user exits.

See the detailed information about the following user exits:

- “The FTCHKCMD user exit” on page 702
- “The FTPOSTPR user exit” on page 703
- “The FTCHKIP user exit” on page 706
- “The FTCHKPWD user exit” on page 707
- “The FTCHKJES user exit” on page 708
- “The FTP server SMF user exit” on page 710

Sample server user exits

You can find sample user exit routines in SEZAINST:

User exit	Location of sample
FTCHKIP	SEZAINST(FTCHKIP)

User exit	Location of sample
FTCHKPWD	SEZAINST(FTCHKPWD)
FTCHKCMD	SEZAINST(FTCHKCMD), SEZAINST(FTCHKCM1), SEZAINST(FTCHKCM2)
FTCHKJES	SEZAINST(FTCHKJES)
FTPOSTPR	SEZAINST(FTPOSTPA), SEZAINST(FTPOSTPR)

The FTCHKCMD user exit

FTCHKCMD is called when the server receives a command to run RETR, STOR, or any other FTP command. The user exit is passed as follows:

- The user ID
- The command
- The command parameters
- The current directory type of MVS or z/OS UNIX
- The file type of SEQ, JES, or SQL
- The current working directory value
- The address of a buffer that can be used to return modified command arguments
- A buffer to hold a 500 reply extension to explain why the exit denied the request
- The socket address structure of the client's control connection
- The socket address structure of the server's control connection
- A buffer containing the session instance identifier
- A 256-byte scratchpad buffer

The exit can accept the command, reject the command, or modify the arguments passed to the command. When the exit rejects the command, the FTP server always replies `500 User Exit denies Userid` *`userid`* `from using Command` *`command`*. If the exit routine places text into the 500 reply extension buffer, the FTP server replies to the client with reply code 500 and the supplied text before it replies `500 User Exit denies Userid userid from using Command command`.

The FTP server sometimes replies to a client with the arguments of the subcommand the client sent to the server. For example, if a client enters SITE FNIDDER=FNAT, a 200 message is returned to the client: `200-Unrecognized parameter 'FNIDDER=FNAT' on site command`. For such replies, the command arguments included are those returned by FTCHKCMD rather than those originally entered by the client.

The following parameter list is passed to FTCHKCMD:

Offset	Value
+0	Pointer to the fullword return code. Return 0 to accept the command or to pass new arguments to the command. Return a nonzero value to reject the command.
+4	Pointer to a word containing the number of following parameters (12).
+8	Pointer to the 8-byte user ID that is logged in.
+12	Pointer to the 8-byte command being entered.

+16 Pointer to a string containing arguments after the command. The first halfword of the string contains the number of characters that follow.

+20 4-byte character string with current directory type: MVS or z/OS UNIX (left-aligned).

+24 4-byte character string with current file type: SEQ, JES, or SQL.

+28 Buffer with current directory value. The first bytes hold length of remaining buffer. This is an 1102-byte output buffer in which to return modified argument strings. The first 2 bytes must be initialized to the length of the returned command string.

+32 1102-byte output buffer in which to return modified argument strings. You can modify the arguments passed to the command by placing the modified arguments in this buffer. The first 2 bytes must be initialized to the length of the returned command string.

+36 Pointer to a 71-byte buffer in which to return a 500 reply extension to be used only when the exit denies the request. The exit can place text in this buffer to explain why it denied the request. If the exit supplies text in this buffer, the server appends this text to the string 500-UX- and sends this reply prior to the reply `500 Userid userid from using Command command`. The buffer is initialized to blanks before each call to FTCHKCMD.

+40 Pointer to a copy of the socket address structure for the client's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.

+44 Pointer to a copy of the socket address structure for the server's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.

+48 Pointer to a buffer containing a 2-byte length followed by the session identifier created by the daemon when the connection was first established for this session. The identifier has a maximum length of 14 bytes and is unique within this instance of the server.

+52 Pointer to a 256-byte scratchpad buffer, which can be used to pass information between user exits. All exits receive a pointer to this buffer except FTCHKIP and FTCHKPWD. FTP does not query or alter the contents of the scratchpad at any time. The extended tracing (DUMP) identifier of the scratchpad is 87. If extended tracing of the scratchpad is requested, the contents are dumped after execution of the user exit.

Restriction: If the exit is used for USER/PASS command processing, the scratchpad buffer should not contain a pointer unless the storage location that is referenced is available to all address spaces.

The FTPOSTPR user exit

FTPOSTPR is called upon completion of the FTP commands RETR, STOR, STOU, APPE, DELE, and RNTO. The user exit is passed as follows:

- The user ID
- The client IP address
- The client port number
- The current directory type
- The length of the parameter string
- The current working directory
- The current file type
- The FTP reply code
- A buffer containing the FTP reply line sent to the client
- The FTP command code
- The current CONDDISP setting
- The file transfer completion code
- Name of the data set or z/OS UNIX file retrieved or stored
- Two words containing the bytes transferred during execution of this command
- The socket address structure of the client's control connection
- The socket address structure of the server's control connection
- A buffer containing the session instance identifier
- A 256-byte scratchpad buffer
- The one-byte description of the confidence level in successful completion of a transfer
- A buffer containing the FTP reply

The user exit can take action based on any of the information passed to it. The close reason code indicates whether the command completed successfully. The scratchpad buffer can be used to communicate information to other exits or the next instance of this exit.

The following parameter list is passed to FTPOSTPR:

Offset	Value
+0	Pointer to the fullword return code. The value is always 0 and is passed only for consistency with other FTP user exits and parameter lists.
+4	Pointer to a word containing the number of following parameters (17).
+8	Pointer to the 8-byte buffer containing the user ID.
+12	Pointer to the 4-byte client IP address. If the client's address is an IPv6 address, this field points to a word containing x'FFFFFFFF' and the passed socket address structure for the client must be used instead. If the client's address is an IPv4 address, either this field or the socket address structure can be used.
+16	Pointer to the 2-byte client port number. Valid only when the 4-byte client IP address is not x'FFFFFFFF'.
+20	Pointer to the 4-byte character string with current directory type: MVS or z/OS UNIX (left-aligned).
+24	Pointer to a buffer containing the current directory value. The first 2 bytes hold the length of the remaining buffer.
+28	Pointer to the 4-character byte field containing the current file type (SEQ, JES, SQL) left-aligned.
+32	Pointer to the 3-character byte field containing the current FTP reply code.

+36 Pointer to buffer containing the last line of the FTP reply. The first 2 bytes contain the length of the remaining buffer.

+40 Pointer to the 4-byte field containing the current FTP command code.

+44 Pointer to the 1-character byte field containing the current CONDDISP setting: C for catalog, D for delete.

+48 Pointer to the 4-byte binary field with close reason code:
- 0 — Transfer completed normally.
- 4 — Transfer completed with errors; see FTP reply code and text string.
- 8 — Transfer completed with socket communication errors; transfer is ended and no response can be sent to client.
- 12 — Transfer aborted after data connection was established.
- 16 — Transfer aborted with SQL file errors after data connection was established.

+52 Pointer to a buffer containing the name of the data set or z/OS UNIX file just retrieved or stored. The first two bytes hold the length of the remainder, and the remainder of the buffer (up to 1023 bytes) holds any additional path specification beyond the current working directory and the file name.

+56 Pointer to two contiguous words containing the bytes transferred during execution of the current FTP command. The first word holds the number of gigabytes transferred. The second word holds the number of bytes transferred in addition to the number of gigabytes transferred. The number of bytes value (word 2) can be up to 4 gigabytes.

+60 Pointer to a copy of the socket address structure for the client's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.

+64 Pointer to a copy of the socket address structure for the server's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.

+68 Pointer to a buffer containing a 2-byte length followed by the session identifier created by the daemon when the connection was first established for this session. The identifier has a maximum length of 14 bytes and is unique within this instance of the server.

+72 Pointer to a 256-byte scratchpad buffer, which can be used to pass information between user exits. All exits receive a pointer to this buffer except FTCHKIP and FTCHKPWD. FTP does not query or alter the contents of the scratchpad at any time. The extended tracing (DUMP) identifier of the scratchpad is 87. If extended tracing of the scratchpad is requested, the contents are dumped after execution of the user exit.

+76 Pointer to a 1-byte description of the confidence level in successful completion of a transfer. Possible values are:

X'00'	Confidence level is High. No errors were detected on the inbound transfer
X'01'	Confidence level is NoEOF due to a missing EOF marker in an inbound file being transferred using STRUCTURE RECORD, MODE B, or MODE C.
X'02'	Confidence level is Low because the client did not respond after the inbound transfer or another error was reported. Low overrides NoEOF if both conditions are present.
X'03'	Confidence level is Unknown because this is an outbound transfer. An outbound transfer reports a confidence level of Low if an error occurs shutting down the data connection. Otherwise, outbound transfers are reported as Unknown even if no error was detected because not all checks can be done for outbound transfers
X'04'	Confidence level checking is not active. See "CHKCONFIDENCE statement (FTP client and server) statement" on page 760.

+80 Pointer to a buffer containing the complete text of the server reply that was sent to the client. The first two bytes contain the length of the remaining buffer.

When the length of the remaining buffer is 0, FTP could not obtain sufficient storage to hold the complete text of the server reply. You can obtain the last line of the reply by inspecting the parameter at offset x'36'.

The FTCHKIP user exit

FTCHKIP is called at the initial stage of login or whenever the user issues an OPEN command to open a new connection. The IP and PORT addresses of the local and remote hosts are passed to the user exit. The user exit can use them to determine whether the remote host's control connection should be canceled. The message `421 User Exit rejects open for connection` is sent to the user if the connection is denied. The following parameter list is passed to FTCHKIP.

Offset	Value
+0	Pointer to the word with the return code
+4	Pointer to a word containing the number of following parameters (7)
+8	Pointer to the fullword remote IP address. If the client's address is an IPv6 address, this field points to a word containing x'FFFFFFFF' and the passed socket address structure for the client must be used instead. If the client's address is an IPv4 address, either this field or the socket address structure can be used.
+12	Pointer to the halfword remote port number. Valid only when the fullword remote IP address is not x'FFFFFFFF'.
+16	Pointer to the fullword local IP address. If the server's address is an IPv6 address, this field points to a word containing x'FFFFFFFF' and the passed socket address structure for the server should be used instead. If the server's address is an IPv4 address, either this field or the socket address structure can be used.
+20	Pointer to the halfword local port number. Valid only when the fullword local IP address is not x'FFFFFFFF'.
+24	Pointer to a copy of the socket address structure for the client's control

connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.

+28 Pointer to a copy of the socket address structure for the server's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.

+32 Pointer to a buffer containing a 2-byte length followed by the session identifier created by the daemon when the connection was first established for this session. The identifier has a maximum length of 14 bytes and is unique within this instance of the server.

FTCHKIP has been placed before the user logs in, and if access is denied, the user receives a message and the control connection is severed. This message comes at a point when most clients expect to continue with the login process by sending the user ID and password. Even though it is possible, some FTP clients might not expect a 421 message at this point, but it is the most appropriate place for this exit.

The FTCHKPWD user exit

The FTP server calls the FTCHKPWD exit after the server receives the user ID and password from the FTP client, but before the server uses the password to authenticate the user logging in. When the user logs in anonymously, and you have coded ANONYMOUSLEVEL 3 in the server's FTP.DATA, and the server prompts the user for an email address, the FTP server calls this exit:

- after the server receives the USER command,
- and again after it receives the PASS command.

The exit has the option of rejecting the attempt to log in to the FTP server. The reply `Login attempt by '<user>' rejected by user exit` is sent to the user if the exit rejects the request to log in to the FTP server. The following parameter list is passed to FTCHKPWD.

Offset	Value
+0	Pointer to the word with the return code **Requirement:** The FTCHKPWD exit routine must set the return code to a nonzero value to reject the attempt to log in. The FTCHKPWD exit routine must set the return code to zero to permit the user to continue logging in.
+4	Pointer to a word containing the number of following parameters (8)
+8	Pointer to the 8-byte ID of the user logging in
+12	Pointer to the 8-byte password of the user logging in **Results:** • The password is passed without alteration to the exit routine when one of the following situations occurs: – You have enabled RACF mixed case password support

  - Your security product is not RACF, but it uses the RACF SAF interface to indicate that mixed passwords are enabled.

  Otherwise, the password is translated to uppercase before it is passed to the exit routine.
- If the user logs in with a password phrase, this parameter is set to the first eight characters of the password phrase.

Tip: The password or entire password phrase is also passed to the exit as the parameter at offset +36.

+16 Pointer to a buffer containing a field that is 2 bytes in length followed by the user data

See "ANONYMOUS (FTP server) statement" on page 738 for more information about this field.

+20 Pointer to a word containing the total number of bad passwords entered during this session

+24 Pointer to a copy of the socket address structure for the client's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.

+28 Pointer to a copy of the socket address structure for the server's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.

+32 Pointer to a buffer containing a field that is 2 bytes in length followed by the session identifier that was created by the daemon when the connection was first established for this session. The identifier has a maximum length of 14 bytes and is unique within this instance of the server.

+36 Pointer to a buffer containing a field that is 2 bytes in length followed by the password or password phrase the user entered to log in to the FTP server.

Notes:
- When PASSPHRASE FALSE is configured in FTP.DATA of the server, the pointed buffer consists of a 2-byte field, which contains zeros, and is followed by 100 blanks.
- When PASSPHRASE TRUE is configured in FTP.DATA of the server, the pointed buffer consists of a 2-byte field, which contains the length of the password or password phrase, and is followed by a 100-byte field. The 100-byte field contains the password or password phrase that is used to log in to FTP and is right-padded with blanks that are up to 100 characters in length.

The FTCHKJES user exit

FTCHKJES is called if the server is in FILETYPE=JES mode and the client tries to submit a job. The exit can allow or refuse the job to be submitted to the JES internal reader based on any criteria passed to the exit. For example, the exit can look for a USER= parameter on the JOB statement and check it against the client's user ID. The reply `550 User Exit refuses this job to be submitted by userid` is sent to the user if the remote job submission is denied. The following parameter list is passed to FTCHKJES.

Offset	Value
+0	Pointer to the word with the return code
+4	Pointer to a word containing the number of following parameters (13)
+8	Pointer to the 8-character user ID that is logged on
+12	Pointer to the buffer containing the current logical record being submitted
+16	Pointer to a word with the number of bytes in the buffer
+20	Pointer to a word containing the JES LRECL being used
+24	Pointer to a word containing the logical record number
+28	Pointer to a word containing the total number of bytes transferred so far
+32	Pointer to a word containing the unique client ID
+36	Pointer to a word containing the JES RECFM (0 for fixed, 1 for variable)
+40	Pointer to a word containing the JES user exit anchor. (One possible use of this anchor is to provide the exit routine with a location to store the address of a persistent storage area for handling multiple calls.)
+44	Pointer to a copy of the socket address structure for the client's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.
+48	Pointer to a copy of the socket address structure for the server's control connection. This area is mapped by the SOCKADDR DSECT found in the sample exit. The FAMILY field denotes whether the structure contains an IPv4 or an IPv6 address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address itself to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address.
+52	Pointer to a buffer containing a 2-byte length followed by the session identifier created by the daemon when the connection was first established for this session. The identifier is unique within this instance of the daemon. It is included in messages written to SYSLOG and can be used similarly by the exit. It is preferred over the similar client ID found at +32 in the parameter list.
+56	Pointer to a 256-byte scratchpad buffer, which can be used to pass information between user exits. All exits receive a pointer to this buffer, except FTCHKIP and FTCHKPWD. FTP does not query or alter the contents of the scratchpad at any time. The extended tracing (DUMP) identifier of the scratchpad is 87. If extended tracing of the scratchpad is requested, the contents are dumped after execution of the user exit.

The return code word is initialized to 0 so the user exit can return without changing it if there is a correct return code. Any other return code denies access to the resource in question.

The FTP server SMF user exit

To access the (preferred) type 119 FTP SMF records, use the system-wide SMF user exits IEFU83 and IEFU85. See z/OS MVS System Management Facilities (SMF) for more information. For the FTP client, IEFU85 SMF exit is invoked, and for the FTP server, IEFU83 SMF exit is invoked.

Restriction: This exit is called for type 118 records only.

Tip: Some FTP type 119 records are available to the NMI SYSTCPSM programming interface. See the information about real time SMF data NMI (SYSTCPSM) record formats in z/OS Communications Server: IP Programmer's Guide and Reference.

Type 118 SMF record types to be written are based on the SMFCONFIG statement in SMF.DATA. The FTP server SMF user exit is called before a matching type 118 SMF record is written to the SYS1.MANx data set. The user exit allows site-specific modifications to the record and can prevent the record's being written to the SYS1.MANx data set.

To enable the exit, include the SMFEXIT statement in the FTP.DATA data set.

Requirement: The routine must be named FTPSMFEX and placed in an installation-defined link library or an APF-authorized data set defined by a STEPLIB DD statement in the FTPD cataloged procedure. FTP calls the SMF user exit before each type 118 SMF record is written.

On entry to FTPSMFEX, register 1 contains a pointer to the following 2-word parameter list:

Offset	**Value**
0	Pointer to the return code
4	Pointer to the type 118 SMF record

Prior to calling the SMF user exit, the return code is set to 0. A return code of 0 specifies that the SMF record is written. To suppress writing the SMF record to the SYS1.MANx data set, the user exit must change the return code to a nonzero value.

FTP client user exits

The FTP client uses user exits to provide security to enable the administrator to control the FTP commands that are sent to the server or to monitor replies that are sent from the FTP server. For example, an administrator can perform the following actions:

- Protect some data sets, which a user has access to, from being transferred from the z/OS host.
- Inspect or modify the names of data sets that are specified on file transfer subcommands by end users.
- End an FTP client address space, if that client is in the process of sending an unauthorized FTP command.
- Inspect each reply from the FTP server and, if certain replies are received, end the client.

Restrictions:

- User exit routines must be in a cataloged data set which is APF-authorized. The data set must be made available to the FTP client via standard z/OS load module search, such as adding the data set to the LNKLIST or via STEPLIB.
- User exit routines must be reentrant.
- User exit routines should be written in Assembler Language. Standard linkage conventions must be used. See z/OS MVS Programming: Assembler Services Guide for the linkage conventions.

  Restriction: User exit routines can be written in C, but C exit routines cannot return an exit reason code. If necessary, you can return a message in the optional message field to display the reason code.
- User exit routines are invoked in TCB mode, program problem state, with AMODE(31). If the user exit routine changes a setting, it must restore it before returning to the caller.
- User exit routines must communicate the result of their processing back to FTP client by setting a return code in register 15 and a reason code in register 0 (in case of a non-zero return code) before returning to the caller.
- FTP client user exits are not supported when the FTP client is invoked in an environment in which the FTP client cannot be executed as an authorized program or command. For example, FTP client user exits are not supported in the dynamic TSO environment that the IKJTSOEV service builds.

The parameter list that is passed to the user exit routine is an array of pointers to values. The first word of the parameter list points to a word that contains the number of parameters that follow. This helps you to handle any future releases that might increase the number of parameters in these parameter lists. The FTP client passes the TCP connection ID to each user exit. The TCP connection ID parameter uniquely identifies a control connection, and remains the same for all user exit calls that are associated with a specific control connection. You can use the TCP connection ID to correlate user exit calls.

To install your user exit routines, associate them with the defined user exit by using either of these methods:

- The EXIT statement of the PROG*xx* parmlib member. The EXIT statement allows an installation to add exit routines to an exit. At IPL, you can use PROG=*xx* to specify the particular PROG*xx* parmlib member that the system is to use. For example, you can specify:
  ```
  EXIT ADD EXITNAME(EZAFCCMD) MODNAME(CSFTPEX1)
  ```
- The SETPROG EXIT operator command. This command performs the same functions as the EXIT statement of the PROGxx parmlib member. For example, you can specify:
  ```
  SETPROG EXIT, ADD, EXITNAME=EZAFCCMD, MODNAME=CSFTPEX1
  ```

For more information about user exits installation, see Exit Routines - Using Dynamic Exit Services in z/OS MVS Programming: Authorized Assembler Services Guide.

Dynamic exit services (DES) allows multiple exit routines to be run when a user exit is called. Multiple user exit routines mean that an exit routine called earlier in the sequence can end the client or the current command before the remaining exit routines are called. In that case, no remaining exit routines are called.

Multiple exit routines also mean that another exit routine might modify your output before the FTP client can apply it, and might modify the input from the FTP client before your exit routine receives it. Use caution when creating tokens or handles based on input parameters to avoid conflicts with other user exit routines.

Restriction: FTP cannot control the call sequence of the multiple exit routines. Do not assume your exit routine is called in any particular sequence, such as first or last.

See the detailed information about the following user exits:
- “The EZAFCCMD user exit”
- “The EZAFCREP user exit” on page 718
- “Using both EZAFCCMD and EZAFCREP user exits” on page 720

Sample client user exits

You can find user exit samples in *hlq*.SEZAINST(EZAFCEXT). EZAFCEXT is a JCL file. It creates user exit routine load modules. When the JCL is submitted, it creates four load modules. EZAFCCMD and EZAFCREP are user exit routines. EZAFCCOM and ASMTSYSL are assistant load modules that are called by the EZAFCCMD and EZAFCREP user exit routines.

Table 43. User exit samples

Load module	**Language**	**Description**
EZAFCCMD	Assembler	EZAFCCMD works in conjunction with the EZAFCREP. These two exits maintain a shared storage area where the FTP session state is maintained. The EZAFCCMD maintains information in the shared session data area based on which commands are being sent to the server. When a file transfer has completed successfully, a message is written to syslogD.
EZAFCREP	Assembler	EZAFCREP works in conjunction with the EZAFCCMD. These two exits maintain a shared storage area where the FTP session state is maintained. EZAFCREP analyzes the replies to the commands that were sent to the server and parses information from a select set of replies. When a file transfer has completed successfully, a message is written to syslogD.
EZAFCCOM	Assembler	EZAFCCOM is called from both of these exits during initialization. EZAFCCOM contains common code for the EZAFCCMD (EZAFCCMD) and EZAFCREP (EZAFCREP) FTP client exit routines.
ASMTSYSL	Assembler	ASMTSYSL provides an interface for writing messages to z/OS syslogD from non-C programs.

The EZAFCCMD user exit

The EZAFCCMD user exit is called by the FTP client before each command is sent to the FTP server. It can be used to inspect an FTP command, modify the arguments of an FTP command, reject an FTP command, or end the FTP client before the command is sent to the server.

The following table is the parameter list that is passed to the EZAFCCMD user exit:

Table 44. Parameter list passed to the EZAFCCMD user exit

+ Offset (decimal)	Value
0	Pointer to a binary fullword that contains the number of parameters that follow (15).
4	Pointer to the 8-byte TCP connection ID. This is a character field. The TCP connection ID parameter identifies a control connection, and remains the same for all user exit calls that are associated with a specific control connection.
8	Pointer to a 256-byte buffer that contains a 2-byte field followed by the remote FTP user ID. The first two bytes hold the length of the remote FTP user ID. The buffer is padded with NULLs.
12	Pointer to an 8-byte buffer that contains the FTP command that is being sent to the server. The buffer is padded with blanks.
16	Pointer to a 5122-byte buffer that contains a 2-byte field followed by the arguments of the FTP command reported at offset 12. The first two bytes hold the length of the FTP command arguments. This buffer is padded with NULLs.
20	A 4-byte character string with the current local FTP client directory type: MVS or z/OS UNIX file system (left-aligned). The buffer is padded with blanks.
24	A 4-byte character string with the local UNIX file type: FILE or FIFO. The buffer is padded with blanks.
28	Pointer to a 1026-byte buffer that contains a 2-byte field followed by the current local directory. The first two bytes of the buffer hold the length of the current local directory. The buffer is padded with NULLs.
32	Pointer to a 1026-byte buffer that contains a 2-byte field followed by the fully qualified local MVS data set name or the absolute path name of the local z/OS UNIX file. The first two bytes hold the length of the MVS data set name or path name. The value of the length is set to zero when the command is not associated with a local MVS data set name or z/OS UNIX file.

Table 44. Parameter list passed to the EZAFCCMD user exit (continued)

+ Offset (decimal)	Value
36	Pointer to a 2-byte length field followed by a buffer containing the local FTP client configuration options (See note 1 following this table). The length field is set to the length of the buffer that follows. The configuration options come from the z/OS FTP client default configuration, the z/OS FTP client START parameters, and the z/OS FTP client FTP.DATA file or data set. Data in the buffer is in the format of *2-byte binary total length2-byte binary lengthconfig_option = config_value*, and is printable EBCDIC apart from the length field. The 2-byte length field does not include the length field itself. When you reach a length field with the value zero, there are no more entries in the buffer. The buffer is updated when a configuration option is changed by the following subcommands: • LOCSITE • MODE • TYPE • STRUCT • ASCII • BIG5 • BINARY • BLOCK • EBCDIC • EUCKANJI • HANGEUL • IBMKANJI • JIS78KJ • JIS83KJ • KSC5601 • SCHINESE • TCHINESE • UCS2 • SJISKANJI • STREAM • RECORD • STRUCTURE • CCC • PROTECT • PROTECT • SAFE • CLEAR • PRIVATE **Requirement:** Do not assume that a specific configuration option is always located at the same offset in this buffer. You must scan the buffer on each call to your exit routine to locate the specific option you are interested in.

Table 44. Parameter list passed to the EZAFCCMD user exit (continued)

+ Offset (decimal)	**Value**
40	Pointer to a copy of the socket address structure for the client control connection. This area is mapped by the SOCKADDR DSECT (See note 2 following this table). The FAMILY field denotes whether the structure contains an IPv4 (Family=2, AF_INET) or an IPv6 (Family=19, AF_INET6) address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address. See note 2 following this table for a sample layout of the socket address structure.
44	Pointer to a copy of the socket address structure for the server control connection. This area is mapped by the SOCKADDR DSECT (See note 2 following this table). The FAMILY field denotes whether the structure contains an IPv4 (Family=2, AF_INET) or an IPv6 (Family=19, AF_INET6) address. When the family is AF_INET, the structure contains an IPv4 address. When the FAMILY is AF_INET6, you must inspect the address to determine whether it is an IPv6 address or an IPv4 mapped IPv6 address. See note 2 following this table for a sample layout of the socket address structure.
48	Pointer to a copy of the socket address structure for the SOCKS server connection when the client connects to the FTP server through a SOCKS server. This area is mapped by the SOCKADDR DSECT (See note 2 following this table). If no SOCKS server is used, the first two bytes of the address structure will be set to X'0000'. The SOCKS server IP address is in the AF_INET address family.
52	Pointer to a 40-byte buffer containing the socket APPLDATA for the current control connection The FTP client APPLDATA for the control connection contains security related information for the control connection. For more information about APPLDATA, see FTP Client Application Programming Interface (API) in z/OS Communications Server: IP Programmer's Guide and Reference.
56	Pointer to a 5122-byte output buffer to return modified command arguments. Your exit routine can modify the arguments that are passed to the command by placing the modified arguments in this buffer. The first two bytes must be initialized to the length of the modified command arguments.
60	Pointer to a 71-byte buffer to return a message when the user exit routine ends the FTP client or rejects a command. Your exit routine can place text in this buffer to explain why it ended the FTP client or rejected the command. This buffer is initialized to the blanks before each call to the EZAFCCMD user exit.

Note:

1. The following sample illustrates how the local configuration options structure is built:

```
H'16',C'OPTION1 = VALUE1',H'18',C'OPTION2 = VALUE234',,,,H'0'

2-bytes binary with the value 16, followed by a 16-byte character string
2-bytes binary with the value 18, followed by an 18-byte character string
2-byte binary with the value zero (no more options in the structure)
```

All entries use a common format with the option name followed by a space, followed by an equal sign (=), followed by another space, followed by the value of the option.

2. Sample socket address structure layout (based upon the SOCKADDR structure in SYS1.MACLIB(BPXYSOCK):

```
*
* --------------------------------------------------------------------
*
* Socket address structure
*
* --------------------------------------------------------------------
*
SOCKADDR DSECT
SOCKLEN  DC    AL1(0)                *If FAM=2, this field is zero
*                                    *If FAM=19, this field is the
*                                    *length of the structure
SOCKFAM  DC    AL1(0)                *Socket family
SOCKIPV4 EQU   2                     *IPv4 socket structure
SOCKIPV6 EQU   19                    *IPv6 socket structure
SOCKPORT DC    AL2(0)                *Port number
IPV4SOCK DS    0C
IPV4ADDR DC    A(0)                  *IPv4 address
         ORG   IPV4SOCK
IPV6FLOW DC    A(0)                  *IPv6 flow label
IPV6ADDR DC    XL16'00'              *IPv6 address
         ORG
```

The EZAFCCMD user exit can return the following return codes to the FTP client in register 15:

RC=0 The exit is to send the command to server.

RC=4 The exit accepts the command but modifies the arguments.

RC=8 The exit rejects the command.

RC=12 The exit is to end the FTP client address space.

The EZAFCCMD user exit can also return a reason code to the FTP client in register 0 if the return code in register 15 is not 0.

Results

- When user exit EZAFCCMD returns code 0, FTP continues processing the command.
- When user exit EZAFCCMD returns code 4,
  - If the arguments of the FTP command are modified, message EZA1532I is displayed.
  - The commands PASS and ACCT have security sensitive data in the command arguments and the arguments are not passed to the user exit. However, the exit is allowed to replace the arguments of the commands.
  - The command ADAT has security sensitive data in the command argument and the argument is not passed to the user exit. Message EZA1545I indicates that the modified argument will be ignored.
  - Because the FTP server and client must remain synchronized, changing the arguments of the following commands leads to unpredictable results:
    - AUTH
    - EPRT
    - EPSV

      - MODE
      - PBSZ
      - PORT
      - PROT
      - REST
      - SITE
      - STRU
      - TYPE
      - XLMT
      - RETR during a load module transfer
      - STOR during a load module transfer

      Therefore, the changed arguments of these commands are ignored and an EZA1545I message is displayed.
  – If more than one exit routine is associated with this user exit, the FTP client passes the modified arguments to the next user exit routine. If this is the only exit routine associated with this user exit, or it is the last exit routine in the calling sequence for this user exit, the FTP client sends the command with modified arguments to the server.
- When user exit EZAFCCMD returns code 8,
  – FTP logs the reason code in register 0 with message EZA1533I, but does not interpret the reason code. The exit routine determines what the reason codes means.
  – If the exit routine returned a message in the 71-byte buffer provided at offset 64 in the EZAFCCMD parameter list, the FTP client displays message EZA1556I before message EZA1533I to explain why the user exit routine rejected the command. The client always displays message EZA1533I.
  – The exit cannot reject the QUIT command, no matter whether the QUIT command is from the QUIT subcommand or from the CLOSE subcommand.
  – If more than one exit routine is associated with this user exit, the FTP client stops calling exit routines for this command.
- When user exit EZAFCCMD returns code 12,
  – FTP logs the reason code in register 0 with message EZA1546I, but does not interpret the reason code . The exit routine determines what the reason codes means.
  – If the exit routine returned a message in the 71-byte buffer provided at offset 64 in the EZAFCCMD parameter list, the client displays message EZA1556I before message EZA1546I to explain why the user exit routine ended the FTP client. The client always displays message EZA1556I.
  – The exit cannot end the FTP client when the QUIT command is from the QUIT subcommand. However, the client always ends as part of QUIT subcommand processing.
  – The exit can end the FTP client when the QUIT command is from the CLOSE subcommand.
  – If more than one exit routine is associated with this user exit, the FTP client stops calling exit routines for this command.

Examples

Example 1: If the EZAFCCM1 user exit routine is installed for the EZAFCCMD user exit, EZAFCCM1 modifies the argument of the CWD command to `'/u/user1'`:

```
EZA1460I Command:
cd 'user1'
EZA1532I User exit EZAFCCMD module EZAFCCM1 modified the FTP command arguments
EZA1701I >>> CWD /u/user1
250 HFS directory /u/user1 is the current working directory.
```

Example 2: If the EZAFCCM2 user exit routine is installed for the EZAFCCMD user exit, EZAFCCM2 rejects the PORT command:

```
EZA1460I Command:
ls
EZA1556I EZAFCCMD message: EZAFCCM2 Rejected the FTP command
EZA1533I User exit EZAFCCMD module EZAFCCM2 prevented user USER1 from sending co
mmand PORT - exit reason code x'00000004' (4)
EZA1636I *** I can't open a data-transfer connection:
EZZ9830I USER13 FTP failed - Cmd = 20(ls) Reply = n/a NX CEE RC = 2720
```

Example 3: If the EZAFCCM3 user exit routine is installed for the EZAFCCMD user exit, EZAFCCM3 ends the FTP client:

```
EZA1460I Command:
get '/u/user1/ftp.example' '/u/user2/ftp.example'
EZA1556I EZAFCCMD message: EZAFCCM3 canelled the FTP client
EZA1546I User exit EZAFCCMD module EZAFCCM3 ended the FTP client - exit reason c
ode x'00000004' (4)
EZA1636I *** I can't open a data-transfer connection:
EZA1701I >>> QUIT
221 Quit command received. Goodbye.
```

Example 4: Multiple user exit routines can be installed for the EZAFCCMD user exit. The EZAFCCM4 and EZAFCCM5 user exit routines are installed for the EZAFCCMD user exit. EZAFCCM4 modifies the argument of the LIST command to `'/u/user2'` and EZAFCCM5 ends the FTP client when it receives the LIST command:

```
EZA1460I Command:
dir '/u/user1'
EZA1701I >>> PORT 9,42,105,183,4,31
200 Port request OK.
EZA1532I User exit EZAFCCMD module EZAFCCM4 modified the FTP command arguments
EZA1556I EZAFCCMD message: EZAFCCM5 cancelled the FTP client
EZA1546I User exit EZAFCCMD module EZAFCCM5 ended the FTP client - exit reason c
ode x'00000004' (4)
EZZ9830I USER11 FTP failed - Cmd = 14(dir) Reply = 200 NX CEE RC = 2814
EZA1701I >>> QUIT
221 Quit command received. Goodbye.
```

The EZAFCREP user exit

The EZAFCREP user exit is called when the FTP client receives a single-line reply or one line of a multiple line reply over the control connection that is sent from the server. A user exit routine that you write for EZAFCREP can inspect the FTP server reply, or end the FTP client after the FTP client receives a certain line of the reply sent from the server.

The following table is the parameter list that is passed to the EZAFCREP user exit:

Table 45. Parameter list passed to the EZAFCREP user exit

+ Offset (decimal)	**Value**
0	Pointer to a binary fullword that contains the number of parameters that follow (4).

Table 45. Parameter list passed to the EZAFCREP user exit (continued)

+ Offset (decimal)	Value
4	Pointer to the 8-byte TCP connection ID. This is a character field. The TCP connection ID parameter uniquely identifies a control connection, and will remain the same for all user exit calls that are associated with a specific control connection.
8	Pointer to a buffer that contains a 2-byte length field followed by the FTP server reply line. The first two bytes hold the length of FTP server reply line. The buffer is padded with NULLs.
12	Pointer to a 40-byte buffer containing the socket APPLDATA for the current control connection. For more information about APPLDATA, see FTP Client Application Programming Interface (API) in z/OS Communications Server: IP Programmer's Guide and Reference.
16	Pointer to a 71-byte buffer to return a message when the user exit routine ends the FTP client. The user exit can put text in this buffer to explain why it ends the FTP client. If the exit supplies text in this buffer, the FTP client displays this text as message EZZ1556I. This buffer is initialized to blanks before each call to the EZAFCREP user exit.

The EZAFCREP user exit can return the following return codes to the FTP client in register 15:

RC=0 The exit accepts the reply.

RC=12 The exit ends the FTP client address space.

It can also return a reason code to the FTP client in register 0 if the return code in register 15 is not 0.

Results

- When the EZAFCREP user exit returns code 0, FTP continues processing the reply.
- When the EZAFCREP user exit returns code 12,
  - FTP logs the reason code in register 0 with message EZA1546I, but does not interpret the reason code. The exit routine determines what the reason codes means.
  - If the exit routine places a message into the 71-byte buffer, the client displays it as message EZA1556I before message EZA1546I. The client always displays message EZA1556I.
  - If the input to the EZAFCREP user exit was one line of a multiple line reply, FTP flushes the remaining lines of the reply from its receive buffer, but does not print them.
  - The FTP client ignores the return code if the EZAFCREP user exit decides to end the FTP client for a reply to the QUIT command.
  - The FTP client ignores the return code if the EZAFCREP user exit decides to end the FTP client for a reply with the reply code in the range 100 - 199.

Example

If the user exit EZAFCREP is active, it is called for each line of a reply as it is received, and FTP displays each line of reply as it is received. If the user exit ends the FTP client in the middle of a multiple line reply, the FTP client will flush all remaining lines of the reply but not print them. The user exit will not be called again for this instance of the FTP client.

Suppose two user exit routines EZAFCRE1 and EZAFCRE2 are both installed for user exit EZAFCREP. EZAFCRE1 ends the FTP client when it receives the reply `'214 – A fully qualified directory name (specified in quotes or'` from subcommand `help server cwd`, EZAFCRE2 accepts any FTP reply. Suppose the operator installs the two exit routines in this order: first EZAFCRE1, then EZAFCRE2.

```
help server cwd
EZA1701I >>> HELP CWD
EZA1582I The foreign server has this help:
214-CWD directory-name: changes the working directory to this directory-name
214-by appending it to the present working directory name.
214-A fully qualified directory name (specified in quotes or
EZA1556I EZAFCREP message: EZAFCRE1 Cancelled the FTP client
EZA1546I User exit EZAFCREP module EZAFCRE1 ended the FTP client - exit reason c
ode x'00000001' (1)
EZA1701I >>> QUIT
221 Quit command received. Goodbye.

The log on the system console:
- 00.07.42  EZAFCRE1 - Input - Offset: +00 Value: 00000003
- 00.07.42  EZAFCRE1 - Input - Offset: +04 Value: 00000088
- 00.07.42  EZAFCRE1 - Input - Offset: +08 Value: 214-CWD directory-
           - name: changes the working directory to this directory-name
- 00.07.42  EZAFCRE2 - Input - Offset: +00 Value: 00000003
- 00.07.42  EZAFCRE2 - Input - Offset: +04 Value: 00000088
- 00.07.42  EZAFCRE2 - Input - Offset: +08 Value: 214-CWD directory-
- name: changes the working directory to this directory-name
- 00.07.42  EZAFCRE1 - Input - Offset: +00 Value: 00000003
- 00.07.42  EZAFCRE1 - Input - Offset: +04 Value: 00000088
- 00.07.42  EZAFCRE1 - Input - Offset: +08 Value: 214-by appending it
           - to the present working directory name.
- 00.07.42  EZAFCRE2 - Input - Offset: +00 Value: 00000003
- 00.07.42  EZAFCRE2 - Input - Offset: +04 Value: 00000088
- 00.07.42  EZAFCRE2 - Input - Offset: +08 Value: 214-by appending it
- to the present working directory name.
- 00.07.42  EZAFCRE1 - Input - Offset: +00 Value: 00000003
- 00.07.42  EZAFCRE1 - Input - Offset: +04 Value: 00000088
- 00.07.42  EZAFCRE1 - Input - Offset: +08 Value: 214-A fully qualified
-  directory name (specified in quotes or
```

In this example, the EZAFCRE1 user exit routine is called three times and the EZAFCRE2 user exit routine is called two times. If EZAFCRE1 ends the FTP client for a certain reply:

- EZAFCRE2 will not be called again for this invocation of EZAFCREP.
- EZAFCREP will not be called again for this FTP client.

The FTP client will flush remaining reply lines from command HELP CWD. The FTP client sends a QUIT command to the server.

Using both EZAFCCMD and EZAFCREP user exits

Results:

- Usually a call to user exit EZAFCCMD will have a corresponding call to user exit EZAFCREP, but not in the following situations:
  - If the EZAFCCMD user exit decides to reject the command or end the FTP client, the command will not be sent to the server, so the EZAFCREP user exit will not be called.

  - You can install more than one EZAFCREP exit routine. If one of the exit routines decides to end the client, the remaining EZAFCREP exit routines will not be called for the corresponding EZAFCCMD call.
- Usually a call to the EZAFCREP user exit will have a corresponding call to user exit EZAFCCMD, but not when the FTP server sends the Good Morning reply 220 to the client before the first command flows to the server from the client. The EZAFCREP user exit is called, but there is no corresponding EZAFCCMD call.

Requirements:
- Your EZAFCCMD exit routine must handle the possibility that your EZAFCREP exit routine will not be called.
- Your EZAFCREP exit routine must handle the possibility that your EZAFCCMD exit was not called.

FTP configuration statements in FTP.DATA

The FTP.DATA configuration data set is optional. The FTP daemon searches for this data set during initialization.

The FTP server search order is:
1. A data set specified by the //SYSFTPD DD statement
2. *ftpserve_job_name*.FTP.DATA
3. /etc/ftp.data
4. SYS1.TCPPARMS(FTPDATA)
5. *hlq*.FTP.DATA data set

As shown in Table 46, the FTP client uses one of the following search orders to obtain the local site parameter values:

Table 46. FTP client search orders

TSO shell	UNIX System Services shell
1. -f	1. -f
2. SYSFTPD DD statement	2. $HOME/ftp.data
3. tso_prefix.FTP.DATA	3. userid.FTP.DATA
4. userid.FTP.DATA	4. /etc/ftp.data
5. /etc/ftp.data	5. SYS1.TCPPARMS(FTPDATA) data set
6. SYS1.TCPPARMS(FTPDATA) data set	6. tcpip_hlq.FTP.DATA file
7. tcpip_hlq.FTP.DATA file	

If you use an MVS data set, this data set should have a logical record length of 80 and a block size that is a multiple of 80. If a UNIX file (such as /etc/ftp.data) is the configuration input, ensure that there are no trailing blanks on the configuration statements, because some specifications might be rejected if trailing blanks are present.

FTP parameters have default values, and you can change these defaults using statements in the FTP.DATA configuration data set. It is not necessary to include all statements in the FTP.DATA data set.

Guideline: Only include the statements if the default value is not what you want, because the default is used for any statement not included in the FTP.DATA data set.

The following names are shipped samples of the FTP.DATA data sets:
- SEZAINST(FTPSDATA) for the server
- SEZAINST(FTCDATA) for the client

The FTP client and server read FTP.DATA once at initialization. Therefore, any changes you make to FTP.DATA are not applied until the next time you start the FTP client and server.

Some FTP server parameters can be changed during an FTP session by issuing the SITE command from the FTP client. Likewise, FTP client parameters can be changed during an FTP session by issuing the subcommand from the FTP client. See the z/OS Communications Server: IP User's Guide and Commands for more information about the SITE command and the locsite subcommand.

Data set attributes play a significant role in FTP performance.

Guidelines: If your environment permits, tune both BLKSIZE and LRECL according to the following guidelines:
- Use a value at or slightly below half of a DASD track as the block size. The half-track threshold for IBM 3380 DASD is 23 476 and for IBM 3390 DASD is 27 998.
- Use FB as the data set allocation format.
- Use cached DASD controllers.
- If your environment permits, use a preallocated data set for FTP transfer operations into MVS.

Summary of FTP client and server configuration statements

The statements for the FTP.DATA data set are summarized in Table 47 on page 723 and explained in detail in “FTP.DATA data set statements” on page 737.

Guidelines:
- Use separate FTP.DATA data sets for the FTP client and the FTP server if you are specifying any conflicting statements.
- When you share the FTP server FTP.DATA data set with the FTP client, understand that some of the values for the statements in the FTP.DATA data set have different meanings in the two environments. If the files are shared, error messages could be generated or values that are not valid could be used for each client using the FTP.DATA data set containing server-only statements.

See “FTP configuration statements in FTP.DATA” on page 721 for more information about the search order for both the client and server.

Table 47. Summary of FTP client and server configuration statements

Statement	Description	Applies to client, server, or both	See
ACCESSERRORMSGS	Allow FTP Server to send detailed login failure replies.	Server	"ACCESSERRORMSGS (FTP server) statement" on page 737
ADMINEMAILADDRESS	Specify a value to use with %E keyword for banner text.	Server	"ADMINEMAILADDRESS (FTP server) statement" on page 738
ANONYMOUS	Allow a remote user to issue USER ANONYMOUS without supplying a logon password.	Server	"ANONYMOUS (FTP server) statement" on page 738
ANONYMOUSFILEACCESS	Specify the type of files (MVS or z/OS UNIX) that anonymous clients are allowed to access.	Server	"ANONYMOUSFILEACCESS (FTP server) statement" on page 741
ANONYMOUSFILETYPEJES	Control access to the FILETYPE SITE keyword of anonymous users when ANONYMOUSLEVEL 3 or greater is specified.	Server	"ANONYMOUSFILETYPEJES (FTP server) statement" on page 742
ANONYMOUSFILETYPESEQ	Control access to the FILETYPE SITE keyword of anonymous users when ANONYMOUSLEVEL 3 or greater is specified.	Server	"ANONYMOUSFILETYPESEQ (FTP server) statement" on page 743
ANONYMOUSFILETYPESQL	Control access to the FILETYPE SITE keyword of anonymous users when ANONYMOUSLEVEL 3 or greater is specified.	Server	"ANONYMOUSFILETYPESQL (FTP server) statement" on page 743
ANONYMOUSFTPLOGGING	Specify whether the FTP server should log FTP session activity for anonymous users.	Server	"ANONYMOUSFTPLOGGING (FTP server) statement" on page 744
ANONYMOUSHFSDIRMODE	Specify the mode bits used for directories created by anonymous users.	Server	"ANONYMOUSHFSDIRMODE (FTP server) statement" on page 745
ANONYMOUSHFSFILEMODE	Specify the mode bits used when storing files created by anonymous users.	Server	"ANONYMOUSHFSFILEMODE (FTP server) statement" on page 746
ANONYMOUSHFSINFO	Specify an anonymous user z/OS UNIX directory information file mask.	Server	"ANONYMOUSHFSINFO (FTP server) statement" on page 747

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
ANONYMOUSLEVEL	Specify the type of anonymous access permitted to users who issue USER ANONYMOUS.	Server	"ANONYMOUSLEVEL (FTP server) statement" on page 748
ANONYMOUSLOGINMSG	Specify anonymous user login messages.	Server	"ANONYMOUSLOGINMSG (FTP server) statement" on page 750
ANONYMOUSMVSINFO	Specify anonymous user MVS information file (LLQ).	Server	"ANONYMOUSMVSINFO (FTP server) statement" on page 751
APPLNAME	Specify the FTP server application name.	Server	"APPLNAME (FTP server) statement" on page 752
ASATRANS	Specify how print control characters should be handled.	Both	"ASATRANS (FTP client and server) statement" on page 753
AUTOMOUNT	Specify whether to mount DASD volumes containing data sets to be accessed.	Both	"AUTOMOUNT (FTP client and server) statement" on page 753
AUTORECALL	Automatically recall data sets migrated by the storage manager.	Both	"AUTORECALL (FTP client and server) statement" on page 754
AUTOTAPEMOUNT	Specify whether to mount tape volumes containing data sets to be accessed.	Both	"AUTOTAPEMOUNT (FTP client and server) statement" on page 755
BANNER	Request that a welcome banner is displayed immediately after a new connection is established.	Server	"BANNER (FTP server) statement" on page 756
BLKSIZE	Specify the block size of newly allocated data sets.	Both	"BLKSIZE (FTP client and server) statement" on page 756
BUFNO	Specify the number of access method buffers.	Both	"BUFNO (FTP client and server) statement" on page 758
CCONNTIME	Defines the amount of time to wait after attempting to close a control connection before terminating it and reporting an error.	Client	"CCONNTIME (FTP client) statement" on page 758
CCTRANS	Specify the SBCS translation table to be used for the control connection.	Client	"CCTRANS (FTP client) statement" on page 758
CCXLATE	Specify the translation table data set for the control connection.	Server	"CCXLATE (FTP server) statement" on page 759

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
CHKCONFIDENCE	Specify that the FTP client or server checks and reports the confidence level in the successful completion of file transfers.	Both	"CHKCONFIDENCE statement (FTP client and server) statement" on page 760
CHKPTINT	Specify the checkpoint interval when FTP is the sending site in a file transfer request.	Both	"CHKPTINT (FTP client and server) statement" on page 762
CHKPTPREFIX	Used to determine the *hlq* for the checkpoint file.	Client	"CHKPTPREFIX (FTP client) statement" on page 764
CIPHERSUITE	Specify the name of a CipherSuite that is used during the TLS handshake.	Both	"CIPHERSUITE (FTP client and server) statement" on page 765
CLIENTERRCODES	Specify whether FTP return codes are to be converted to client error codes.	Client	"CLIENTERRCODES (FTP client) statement" on page 767
CONDDISP	Specify whether FTP should keep or delete a new data set or file when a file transfer ends prematurely.	Both	"CONDDISP (FTP client and server) statement" on page 768
CTRLCONN	Specify code set to be used for the control connection.	Both	"CTRLCONN (FTP client and server) statement" on page 769
DATACLASS	Specify the SMS-managed data class as defined by your organization for FTP.	Both	"DATACLASS (FTP client and server) statement" on page 770
DATACTTIME	Specify the amount time that the client waits after attempting to send or receive data before terminating the connection and reporting an error to the user.	Client	"DATACTTIME (FTP client) statement" on page 772
DATAKEEPALIVE	Specify the data connection keepalive timer.	Both	"DATAKEEPALIVE (FTP client and server) statement" on page 773
DATATIMEOUT	Specify the time that the server waits for a response to a send or for the completion of a passive connection.	Server	"DATATIMEOUT (FTP server) statement" on page 774

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	**Description**	**Applies to client, server, or both**	**See**
DB2	Specify the name of the DB2 subsystem.	Both	"DB2 (FTP client and server) statement" on page 774
DB2PLAN	Specify the name of the DB2 plan to be used by FTP.	Both	"DB2PLAN (FTP cilent and server) statement" on page 775
DBSUB	Specify whether substitution is allowed for double-byte file data that cannot be translated.	Both	"DBSUB (FTP client and server) statement" on page 776
DCBDSN	Specify a data set to be used as a model for allocation of new data sets.	Both	"DCBDSN (FTP client and server) statement" on page 776
DCONNTIME	Specify the amount of time to wait after attempting to close a data transfer before terminating the connection and reporting an error.	Both	"DCONNTIME (FTP client and server) statement" on page 777
DEBUG	Specify to activate a specific trace type.	Both	"DEBUG (FTP client and server) statement" on page 778
DEBUGONSITE	Specify whether an FTP client is allowed to enter the SITE DEBUG command to change general tracing options.	Server	"DEBUGONSITE (FTP server) statement" on page 780
DEST	Specify the NJE destination to which the files are routed when you enter a PUT subcommand.	Server	"DEST (FTP server) statement" on page 780
DIRECTORY	Specify the number of directory blocks to be allocated for the directory of a PDS.	Both	"DIRECTORY (FTP client and server) statement" on page 781
DIRECTORYMODE	Specify how to treat the data set qualifiers below the current directory.	Both	"DIRECTORYMODE (FTP client and server) statement" on page 782
DSNTYPE	Specify the data set name type for new physical sequential data sets.	Both	"DSNTYPE (FTP client and server) statement" on page 783

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
DSWAITTIME	Specify the number of minutes that FTP tries to access an MVS data set that could not be obtained because another job or process was holding the data set.	Both	"DSWAITTIME (FTP client and server) statement" on page 784
DSWAITTIMEREPLY	Specify how often to send the following reply message to the client while the FTP server is waiting for access to an MVS data set.	Server	"DSWAITTIMEREPLY (FTP server) statement" on page 785
DUMP	Specify to activate an extended trace dump ID.	Both	"DUMP (FTP client and server) statement" on page 786
DUMPONSITE	Specify whether an FTP client is allowed to enter the SITE DUMP command to change the extended tracing options.	Server	"DUMPONSITE (FTP server) statement" on page 787
EATTR	Specify whether new data sets can have extended attributes and whether the data sets can reside in the EAS.	Both	"EATTR (FTP client and server) statement" on page 788
EMAILADDRCHECK	Control the extent to which the FTP server validates e-mail addresses entered by FTP clients while logging in to the FTP server.	Server	"EMAILADDRCHECK (FTP server) statement" on page 789
ENCODING	Specify the type of data encoding on the network.	Both	"ENCODING (FTP client and server) statement" on page 790
EPSV4	Direct the FTP client to use EPSV and EPRT commands on IPv4 sessions.	Client	"EPSV4 (FTP client) statement" on page 791
EXTENSIONS	Enable FTP to recognize extensions to FTP that are not described in RFC 959.	Both	"EXTENSIONS (FTP client and server) statement" on page 792
FIFOIOTIME	Specify the FIFOIOTIME statement to set a timeout for reads and writes to a z/OS UNIX named pipe.	Both	"FIFOIOTIME (FTP client and server) statement" on page 794

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
FIFOOPENTIME	Specify the FIFOOPENTIME statement to define the length of time that FTP waits after attempting to open a z/OS UNIX named pipe before reporting an error.	Both	"FIFOOPENTIME (FTP client and server) statement" on page 795
FILETYPE	Specify the operational mode of FTP.	Both	"FILETYPE (FTP client and server) statement" on page 796
FTPKEEPALIVE	Specify the control connection keepalive timer value in seconds.	Both	"FTPKEEPALIVE (FTP client and server) statement" on page 797
FTPLOGGING	Specify whether the FTP server logs FTP session activity for unknown users (that is, users that are not anonymous users).	Server	"FTPLOGGING (FTP server) statement" on page 797
FWFRIENDLY	Specify how data connections are to be set up between the client and the server.	Client	"FWFRIENDLY (FTP client) statement" on page 799
HFSINFO	Specify a file containing welcome messages specific to each FTP server directory visited by an FTP client.	Server	"HFSINFO (FTP server) statement" on page 799
INACTIVE	Set the inactivity timer to a specified number of seconds.	Server	"INACTIVE (FTP Server) statement" on page 800
INACTTIME	Specify the amount of time to wait for an expected response from the server, on either the control or the data connection, before closing the session. Data transfer times that exceed this value does not cause session termination unless the time between data packet arrivals exceeds this value.	Client	"INACTTIME (FTP client) statement" on page 801
ISPFSTATS	Allow FTP to create and maintain statistics for partitioned data set members.	Both	"ISPFSTATS (FTP client and server) statement" on page 802

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
JESENTRYLIMIT	Specify how many JES entries can be displayed at one time with the LIST or NLST command.	Server	“JESENTRYLIMIT (FTP server) statement” on page 802
JESGETBYDSN	Specify how to treat the foreign file name on a GET command when FILETYPE=JES is specified.	Server	“JESGETBYDSN (FTP server) statement” on page 803
JESINTERFACELEVEL	Specify the JES interface level.	Server	“JESINTERFACELEVEL (FTP server) statement” on page 804
JESLRECL	Specify the record length of the job being submitted.	Server	“JESLRECL (FTP server) statement” on page 806
JESPUTGETTO	Specify the number of seconds for the JES PutGet timeout.	Server	“JESPUTGETTO (FTP server) statement” on page 807
JESRECFM	Specify the record format of the job being submitted.	Server	“JESRECFM (FTP server) statement” on page 808
KEYRING	Define the key ring that contains the certificate to be used during the TLS handshake.	Both	“KEYRING (FTP client and server) statement” on page 808
LISTLEVEL	Specifies the format of the LIST command reply.	Server	“LISTLEVEL (FTP server) statement” on page 809
LISTSUBDIR	Specify whether subdirectories of the parent directory are listed when FTP generates a list of files.	Both	“LISTSUBDIR (FTP client and server) statement” on page 810
LOGCLIENTERR	Specify to activate client error logging feature.	Client	“LOGCLIENTERR (FTP client) statement” on page 812
LOGINMSG	Specify the file containing messages to be displayed to FTP clients when they have successfully logged in.	Server	“LOGINMSG (FTP server) statement” on page 813
LRECL	Specify the size of the records in a data set.	Both	“LRECL (FTP client and server) statement” on page 813
MBDATACONN	Specify the multibyte data translation code pages for data connections.	Both	“MBDATACONN (FTP client and server) statement” on page 815
MBREQUIRELASTEOL	Specify whether FTP requires the last record of incoming multibyte files to end with the FTP standard EOL sequence.	Both	“MBREQUIRELASTEOL (FTP client and server) statement” on page 816

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
MBSENDEOL	Specify to the FTP client or server what EOL sequence to use when the ENcoding value is MBCS.	Both	"MBSENDEOL statement (FTP client and server) statement" on page 817
MGMTCLASS	Specify the SMS management class to be assigned to newly allocated data sets.	Both	"MGMTCLASS (FTP client and server) statement" on page 818
MIGRATEVOL	Specify the volume ID for migrated data sets not under the control of IBM Storage Management Systems.	Both	"MIGRATEVOL (FTP client and server) statement" on page 819
MVSINFO	Specify the MVS data sets whose contents are to be returned to the FTP client and displayed to the end user when a user changes directories.	Server	"MVSINFO (FTP server) statement" on page 820
MVSURLKEY	Specify a token that users can enter as part of an FTP URL to encode an MVS data set name.	Server	"MVSURLKEY (FTP server) statement" on page 820
MYOPENTIME	Specify the amount of time to wait for a session to open before terminating the attempt and reporting an error.	Client	"MYOPENTIME (FTP client) statement" on page 821
NETRCLEVEL	Specify how the FTP client searches the NETRC data set for FTP server hostnames.	Client	"NETRCLEVEL (FTP client) statement" on page 821
NONSWAPD	Specify whether the FTP daemon is swappable.	Server	"NONSWAPD (FTP server) statement" on page 822
PASSIVEDATACONN	Specify to direct the server to verify the peer IP address of the data socket is the client's IP address.	Server	"PASSIVEDATACONN (FTP server) statement" on page 823
PASSIVEDATAPORTS	Specify a range of port numbers for the FTP server to use as listening data socket ports.	Server	"PASSIVEDATAPORTS (FTP server) statement" on page 824
PASSIVEIGNOREADDR	Specify to direct the FTP client to ignore the IP address returned from the server on the PASV reply on IPv4 sessions.	Client	"PASSIVEIGNOREADDR (FTP client) statement" on page 824

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
PDSTYPE	Specify the type of MVS directories (PDS or PDSE) FTP should allocate.	Both	"PDSTYPE (FTP client and server) statement" on page 826
PORTCOMMAND	Specify whether the PORT and EPRT commands are accepted or rejected.	Server	"PORTCOMMAND (FTP server) statement" on page 827
PORTCOMMANDIPADDR	Specify the server to accept only PORT or EPRT commands whose IP address matches that of the client.	Server	"PORTCOMMANDIPADDR (FTP server) statement" on page 828
PORTCOMMANDPORT	Specify what range or port values the server accepts as a parameter for the PORT and EPRT commands.	Server	"PORTCOMMANDPORT (FTP server) statement" on page 828
PORTOFENTRY4	Specify the port of entry resource class to use for IPv4 login clients.	Server	"PORTOFENTRY4 (FTP server) statement" on page 829
PRIMARY	Specify the number of tracks, blocks, or cylinders for primary allocation.	Both	"PRIMARY (FTP client and server) statement" on page 830
PROGRESS	Specify the interval between progress report messages generated by the FTP client during an inbound or outbound file transfer.	Client	"PROGRESS (FTP client) statement" on page 831
QUOTESOVERRIDE	Specify use of single quotation marks in file name.	Both	"QUOTESOVERRIDE (FTP client and server) statement" on page 831
RDW	Specify whether RDWs are discarded upon retrieval.	Both	"RDW (FTP client and server) statement" on page 832
RECFM	Specify the record format of a data set.	Both	"RECFM (FTP client and server) statement" on page 833
REMOVEINBEOF	Remove UNIX EOF on inbound ASCII transfers.	Both	"REMOVEINBEOF (FTP client and server) statement" on page 835
REPLY226 (FTP server)	Direct the FTP server to reply to the FTP client with reply code 226 instead of reply code 250.	Server	"REPLY226 (FTP server) statement" on page 835
REPLYSECURITYLEVEL	Specify level of secure information returned in FTP replies.	Server	"REPLYSECURITYLEVEL (FTP server) statement" on page 836

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
RESTGET	Specify whether the checkpoint data set is opened for a GET request.	Client	"RESTGET (FTP client) statement" on page 837
RESTPUT	Specify whether the server supports checkpoint and restart processing when receiving data (put operation).	Server	"RESTPUT (FTP server) statement" on page 838
RETPD	Specify the number of days a newly allocated data set should be retained.	Both	"RETPD (FTP client and server) statement" on page 838
SBDATACONN	Specify single-byte data translation for the data connection.	Both	"SBDATACONN (FTP client and server) statement" on page 840
SBSENDEOL	Specify to the FTP client or server what end of line (EOL sequence to use for outbound ASCII file transfer when the ENcoding value is SBCS.	Both	"SBSENDEOL statement (FTP client and server) statement" on page 841
SBSUB	Specifies whether a substitution is allowed for a data byte that cannot be translated.	Both	"SBSUB (FTP client and server) statement" on page 843
SBSUBCHAR	Specifies the single-byte substitution character for untranslatable data characters.	Both	"SBSUBCHAR (FTP client and server) statement" on page 844
SBTRANS	Specify the SBCS translation table to be used for the data connection.	Client	"SBTRANS (FTP client) statement" on page 844
SECONDARY	Specify the number of tracks, blocks, or cylinders for secondary allocation.	Both	"SECONDARY (FTP client and server) statement" on page 845
SECURE_CTRLCONN	Specify the SECURE_CTRLCONN statement to specify the minimum level of security allowed for the control connection.	Both	"SECURE_CTRLCONN (FTP client and server) statement" on page 846

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
SECURE_DATACONN	Specify the SECURE_DATACONN statement to specify the minimum level of security required on the data connection.	Both	"SECURE_DATACONN (FTP client and server) statement" on page 847
SECURE_FTP	Specify the SECURE_FTP statement to specify whether authentication is required.	Both	"SECURE_FTP (FTP client and server) statement" on page 849
SECURE_HOSTNAME	Specify the SECURE_HOSTNAME statement to specify whether the client verifies the host name in the server's certificate.	Client	"SECURE_HOSTNAME (FTP client) statement" on page 851
SECUREIMPLICITZOS	Specify when the security of the session should be negotiated for implicit TLS connections.	Both	"SECUREIMPLICITZOS (FTP client and server) statement" on page 851
SECURE_LOGIN	Specify the SECURE_LOGIN statement to set the authorization level required for users.	Server	"SECURE_LOGIN (FTP server) statement" on page 852
SECURE_MECHANISM	Specifies which security mechanism the client uses.	Client	"SECURE_MECHANISM (FTP client) statement" on page 854
SECURE_PASSWORD	Specify whether a password is required by the FTP server for an TLS protected session.	Server	"SECURE_PASSWORD (FTP server) statement" on page 855
SECURE_PASSWORD_KERBEROS	Specify whether a password is required for a Kerberos protected session.	Server	"SECURE_PASSWORD_KERBEROS (FTP server) statement" on page 856
SECURE_PBSZ	Specify the maximum size of the encoded data blocks sent during file transfer.	Both	"SECURE_PBSZ (FTP client and server) statement" on page 858
SEQNUMSUPPORT	Specify that sequence numbers in files designated by the ddname INPUT are ignored.	Client	"SEQNUMSUPPORT (FTP client) statement" on page 859
SMF	Specify the default SMF record subtype for all SMF records.	Server	"SMF (FTP server) statement" on page 860

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
SMFAPPE	Specify the SMF record subtype for the APPEND subcommand.	Server	"SMFAPPE (FTP server) statement" on page 862
SMFDCFG	Specify a type 119 SMF record of subtype 71 is collected for the FTP daemon configuration information when the FTP daemon starts.	Server	"SMFDCFG (FTP server) statement" on page 863
SMFDEL	Specify the SMF record subtype for the DELETE subcommand.	Server	"SMFDEL (FTP server) statement" on page 864
SMFEXIT	Call the FTPSMFEX user exit routine.	Server	"SMFEXIT (FTP server) statement" on page 865
SMFJES	Collect SMF records when FILETYPE is JES.	Server	"SMFJES (FTP server) statement" on page 866
SMFLOGN	Specify the SMF record subtype when recording logon failures.	Server	"SMFLOGN (FTP server) statement" on page 867
SMFREN	Specify the SMF record subtype for the RENAME subcommand.	Server	"SMFREN (FTP server) statement" on page 868
SMFRETR	Specify the SMF record subtype for the RETR subcommand.	Server	"SMFRETR (FTP server) statement" on page 869
SMFSQL	Collect SMF records when FILETYPE is SQL.	Server	"SMFSQL (FTP server) statement" on page 870
SMFSTOR	Specify the SMF record subtype for the STOR and STOU subcommands.	Server	"SMFSTOR (FTP server) statement" on page 871
SOCKSCONFIGFILE	Specify the SOCKS server configuration file the FTP client uses to determine which FTP servers require SOCKS protocols.	Client	"SOCKSCONFIGFILE (FTP client) statement" on page 872
SPACETYPE	Specify whether newly allocated data sets are allocated in blocks, cylinders, or tracks.	Both	"SPACETYPE (FTP client and server) statement" on page 873
SPREAD	Specify output in spreadsheet format when file type is SQL.	Both	"SPREAD (FTP client and server) statement" on page 874
SQLCOL	Specify the column headings of the output file.	Both	"SQLCOL (FTP client and server) statement" on page 874

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
SSLV3	Control whether SSLV3 is enabled for connections that are secured using TLS implemented by FTP.	Both	"SSLV3 (FTP client and server connection) statement" on page 875
STARTDIRECTORY	Specify which file system is used initially when a new user logs in.	Server	"STARTDIRECTORY (FTP server) statement" on page 876
STORCLASS	Specify the SMS-managed storage class for the FTP server.	Both	"STORCLASS (FTP client and server) statement" on page 877
SUPPRESSIGNOREWARNINGS	Instruct FTP not to issue message EZYFT47I whenever it ignores a statement coded in FTP.DATA	Both	"SUPPRESSIGNOREWARNINGS (FTP client and server) statement" on page 877
TAPEREADSTREAM	Specify whether to use a more efficient read path (read as stream) to retrieve tape data sets from the server.	Server	"TAPEREADSTREAM (FTP server) statement" on page 878
TLSMECHANISM	Specify how TLS security is implemented.	Both	"TLSMECHANISM (FTP client and server) statement" on page 879
TLSPORT	Set the secure port on which the FTP client or the FTP server implicitly protects the FTP session with TLS.	Both	"TLSPORT (FTP client and server) statement" on page 880
TLSRFCLEVEL	Specify the level of RFC 4217 (*Securing FTP with TLS*) that FTP supports.	Both	"TLSRFCLEVEL (FTP client and server) statement" on page 880
TLSTIMEOUT	Specify the maximum time between full TLS handshakes.	Both	"TLSTIMEOUT (FTP client and server) statement" on page 882
TRACE	Start tracing.	Both	"TRACE (FTP client and server) statement" on page 882
TRACECAPI	Define a control for tracing for a user-written program that uses the callable API interface for the z/OS FTP client.	Client	"TRACECAPI (FTP client) statement" on page 883
TRAILINGBLANKS	Include trailing blanks in fixed format data sets when retrieved.	Both	"TRAILINGBLANKS (FTP client and server) statement" on page 883
TRUNCATE	Allow truncating records that are longer than LRECL.	Both	"TRUNCATE (FTP client and server) statement" on page 884

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
UCOUNT	Specify the unit count for new data set allocations.	Both	"UCOUNT (FTP client and server) statement" on page 885
UCSHOSTCS	Specify the EBCDIC code set to be used for data conversion to or from Unicode.	Both	"UCSHOSTCS (FTP client and server) statement" on page 886
UCSSUB	Specify whether Unicode-to-EBCDIC conversion should use the EBCDIC substitution character or cause the data transfer to be terminated if a Unicode character cannot be converted to a character in the target EBCDIC code set.	Both	"UCSSUB (FTP client and server) statement" on page 886
UCSTRUNC	Specify whether the transfer of Unicode data should be aborted if truncation occurs at the MVS host.	Both	"UCSTRUNC (FTP client and server) statement" on page 887
UMASK	Specify the file mode creation mask.	Both	"UMASK (FTP client and server) statement" on page 887
UNICODEFILESYSTEMBOM	Specify whether to add a Byte Order Mark (BOM) to a file stored in the local file system when the file system code page is UNICODE.	Both	"UNICODEFILESYSTEMBOM (FTP client and server) statement" on page 888
UNITNAME	Specify the unit type for allocation of new data sets.	Both	"UNITNAME (FTP client and server) statement" on page 890
UNIXFILETYPE	Specify the UNIXFILETYPE statement in the FTP server and client to indicate whether to treat z/OS UNIX file system files as regular files or as z/OS UNIX named pipes during file transfer.	Both	"UNIXFILETYPE (FTP client and server) statement" on page 891
VCOUNT	Specify the volume count for allocation of new data sets.	Both	"VCOUNT (FTP client and server) statement" on page 892

Table 47. Summary of FTP client and server configuration statements (continued)

Statement	Description	Applies to client, server, or both	See
VERIFYUSER	Specify whether the FTP server should verify whether a user attempting to log into FTP has been granted access to the server's port profile in the SERVAUTH class.	Server	"VERIFYUSER (FTP server) statement" on page 893
VOLUME	Specify the volume serial number or numbers for allocation of new data sets.	Both	"VOLUME (FTP client and server) statement" on page 895
WRAPRECORD	Specify whether data is wrapped or truncated if no new-line character is encountered before the logical record length is reached.	Both	"WRAPRECORD (FTP client and server) statement" on page 896
WRTAPEFASTIO	Allow write to tape of ASCII stream data to use BSAM I/O routines.	Both	"WRTAPEFASTIO (FTP client and server) statement" on page 896
XLATE	Specify the translation table data set for the data connection.	Server	"XLATE (FTP server) statement" on page 897

FTP.DATA data set statements

These topics cover, in detail, the statements you can use in the FTP.DATA data set. Each statement heading identifies whether the statement applies to FTP client, server, or both.

ACCESSERRORMSGS (FTP server) statement

Use the ACCESSERRORMSGS statement to allow FTP server to send detailed login failure replies to an FTP client.

Syntax

```
       ┌─ACCESSERRORMSGS FALSE──────┐
►►─────┴─ACCESSERRORMSGS──┬───────┬─┴──────────────────────────────────────►◄
                          ├─FALSE─┤
                          └─TRUE──┘
```

Parameters

FALSE
Do not send detailed login failure replies to an FTP client.

TRUE
Send detailed login failure replies to an FTP client.

Examples

To send detailed login failure replies to an FTP client, use the following code:

```
ACCESSERRORMSGS TRUE
```

Usage notes

The text of detailed login failure replies can be traced using the ACC parameter of the DEBUG statement. You do not need to code ACCESSERRORMSGS TRUE to trace this information.

Related topics

- "DEBUG (FTP client and server) statement" on page 778

ADMINEMAILADDRESS (FTP server) statement

Use the ADMINEMAILADDRESS statement to specify a value to substitute for the %E keyword used for the data set or file specified in the BANNER, LOGINMSG, ANONYMOUSMVSINFO, ANONYMOUSLOGINMSG, HFSINFO, and MVSINFO statements. This statement is used to specify the e-mail address of the FTP server administrator.

Syntax

```
►►──ADMINEMAILADDRESS──value────────────────────────────────────────────►◄
```

Parameters

value
The e-mail address displayed when %E is used in BANNER, LOGINMSG, ANONYMOUSMVSINFO, ANONYMOUSLOGINMSG, HFSINFO, and MVSINFO displays.

Examples

```
ADMINEMAILADDRESS TheWebMaster@Myhost.MyCompany.Com
```

Related topics

- "ANONYMOUSHFSINFO (FTP server) statement" on page 747
- "ANONYMOUSMVSINFO (FTP server) statement" on page 751
- "ANONYMOUSLOGINMSG (FTP server) statement" on page 750
- "BANNER (FTP server) statement" on page 756
- "HFSINFO (FTP server) statement" on page 799
- "LOGINMSG (FTP server) statement" on page 813
- "MVSINFO (FTP server) statement" on page 820

ANONYMOUS (FTP server) statement

Use the ANONYMOUS statement to allow remote users to log in as anonymous users.

You can use ANONYMOUSLEVEL, ANONYMOUSFILEACCESS, ANONYMOUSFILETYPESQL, ANONYMOUSFILETYPEJES, and ANONYMOUSFILETYPESEQ in conjunction with ANONYMOUSLEVEL 3 to restrict anonymous users' access to data sets and files. Use ANONYMOUSMVSINFO, ANONYMOUSLOGINMSG, ANONYMOUSHFSINFO, and EMAILADDRCHECK to customize the FTP session for anonymous users.

Requirement: If you choose an ANONYMOUSLEVEL value greater than 1, and you choose STARTDIRECTORY HFS, you must create an anonymous directory structure in the z/OS UNIX. For more information about configuring anonymous logins, see z/OS Communications Server: IP Configuration Guide.

Syntax

```
►►──ANONYMOUS──┬─────────────────────┬──────────────────────────────────►◄
               ├─user_id─────────────┤
               ├─user_id/password────┤
               └─user_id/SURROGATE───┘
```

Parameters

`user_id`
: The security access facility (SAF) identity of the anonymous user. When a remote user enters ANONYMOUS as a user ID, the FTP server treats the login request as though the specified *user_id* was entered instead of ANONYMOUS. The user is prompted for the password to *user_id*. If the user enters the correct password or password phrase, the user is logged in as the specified *user_id*.

  If you are using RACF, the system builds a user accessor environment element (ACEE), and the ANONYMOUS user has access to any resources available to the specified user ID.

`user_id/password`
: The security access facility (SAF) identity and password the FTP server uses for anonymous user. When a remote user enters ANONYMOUS as the user ID, the FTP server treats the login request as though the specified *user_id* was entered instead of ANONYMOUS. The FTP server automatically provides the *password* for the specified *user_id* and the user is logged in as the specified *user_id*. If you are using RACF, the system builds the user ACEE for the specified *user_id* and the ANONYMOUS user has authorized access to the same resources as the specified *user_id*.

  If ANONYMOUSLEVEL 3 is specified, the behavior is different. See "ANONYMOUSLEVEL (FTP server) statement" on page 748 for details.

  Restriction: Do not code a password phrase as *password*.

`user_id/SURROGATE`
: Allows a remote user to enter ANONYMOUS as a user ID. When ANONYMOUS is entered as the user ID, the FTP server treats the login request as though the specified user_ID was entered instead of ANONYMOUS. The FTP Server calls RACF and checks if this user_ID is allowed to login without a password or password phrase.

  Requirement: In order to use this option, ANONYMOUSLEVEL must be greater or equal to 3. See "ANONYMOUSLEVEL (FTP server) statement" on page 748 for details.

Examples

Allow a remote user to enter ANONYMOUS as a user ID and be connected to the server system with the user ID of TERMABC:

```
ANONYMOUS TERMABC/ILLBBACK
```

Tip:

- If you code ANONYMOUSLEVEL 3 in FTP.DATA, you can code additional statements to configure ANONYMOUS support and security. See **Related topics** for more information.

Requirements:

- If you specify a user ID on the ANONYMOUS statement, that user ID must be defined and have a z/OS UNIX segment defined or set to the default value.
- If you code the ANONYMOUS statement without a user ID, the user ID ANONYMO must be defined and must have a z/OS UNIX segment defined or set to the default value.

Results:

- If you code the ANONYMOUS statement without a user ID:
  - The end user is not prompted for a password.
  - If you are using the FTCHKPWD user exit,
    - the exit is called with user ID ANONYMO and password *.
    - If ANONYMOUSLEVEL 3 is coded in FTP.DATA and the FTP server prompts the FTP client for an email address, the email address is passed to the exit as the userdata parameter.
  - The user ID ANONYMO and the STARTDIRECTORY statement in FTP.DATA determine the initial working directory. See initial working directory considerations at the z/OS FTP server in the z/OS Communications Server: IP User's Guide and Commands for more information.
  - The initial working directory is ANONYMO when the STARTDIRECTORY MVS statement is coded in FTP.DATA.
  - The initial working directory is the home directory for the ANONYMO user ID when the STARTDIRECTORY HFS statement is coded in FTP.DATA.
  - If you are using RACF, a user who logs in as 'anonymous' has access to any resources accessible to the ANONYMO user ID.
- If you code the ANONYMOUS statement with a user ID, the user ID you coded and the STARTDIRECTORY statement determine the initial working directory. See initial working directory considerations at the z/OS FTP server in the z/OS Communications Server: IP User's Guide and Commands for more information.
- There is no default for ANONYMOUS. If you do not code the ANONYMOUS statement in FTP.DATA, users are not allowed to log in anonymously.
- See z/OS Communications Server: IP Configuration Guide for more information about anonymous FTP logins.

Related topics:

- "ANONYMOUSLEVEL (FTP server) statement" on page 748
- "ANONYMOUSFILEACCESS (FTP server) statement" on page 741
- "ANONYMOUSFILETYPEJES (FTP server) statement" on page 742
- "ANONYMOUSHFSFILEMODE (FTP server) statement" on page 746
- "ANONYMOUSHFSDIRMODE (FTP server) statement" on page 745

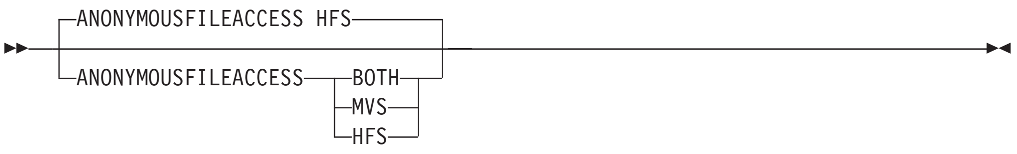
- “ANONYMOUSHFSINFO (FTP server) statement” on page 747
- “ANONYMOUSLOGINMSG (FTP server) statement” on page 750
- “ANONYMOUSMVSINFO (FTP server) statement” on page 751
- “ANONYMOUSFILETYPESEQ (FTP server) statement” on page 743
- “ANONYMOUSFILETYPESQL (FTP server) statement” on page 743
- “EMAILADDRCHECK (FTP server) statement” on page 789
- “STARTDIRECTORY (FTP server) statement” on page 876
- “The FTCHKPWD user exit” on page 707

ANONYMOUSFILEACCESS (FTP server) statement

Use ANONYMOUSFILEACCESS to set the type of files (MVS, z/OS UNIX, or both) that anonymous users are allowed to access. If STARTDIRECTORY is HFS and ANONYMOUSFILEACCESS is HFS, the anonymous user is not allowed to access MVS data sets. If STARTDIRECTORY is MVS and ANONYMOUSFILEACCESS is MVS, the anonymous user is not allowed to access z/OS UNIX files. If STARTDIRECTORY and ANONYMOUSFILEACCESS contradict each other, the anonymous user is not allowed to log in (the login fails). A value of BOTH allows the anonymous user to switch back and forth between MVS and z/OS UNIX data sets.

Restriction: ANONYMOUSFILEACCESS is valid only when ANONYMOUSLEVEL 3 or greater is specified.

Syntax



Parameters

BOTH
Allows anonymous users to access both z/OS UNIX and MVS.

MVS
Allows anonymous users access to only MVS data sets.

HFS
Allows anonymous users access to only z/OS UNIX data sets. This is the default.

Examples

Allow the anonymous users to access both MVS and z/OS UNIX files:

```
ANONYMOUSFILEACCESS BOTH
```

Usage notes

ANONYMOUSFILEACCESS is valid only when ANONYMOUSLEVEL 3 is specified.

Related topics

- "ANONYMOUS (FTP server) statement" on page 738
- "ANONYMOUSLEVEL (FTP server) statement" on page 748
- "ANONYMOUSHFSDIRMODE (FTP server) statement" on page 745
- "ANONYMOUSHFSFILEMODE (FTP server) statement" on page 746
- "STARTDIRECTORY (FTP server) statement" on page 876

ANONYMOUSFILETYPEJES (FTP server) statement

Use the ANONYMOUSFILETYPEJES statement to control the access of anonymous users to the FTP server running in JES operation mode (FILETYPE=JES).

Restriction: The ANONYMOUSFILETYPEJES statement is recognized only when ANYMOUSLEVEL 3 or greater is specified.

Syntax

```
   ┌─ANONYMOUSFILETYPEJES FALSE─────────┐
►►─┴─ANONYMOUSFILETYPEJES──┬─FALSE─┬─┴────────────────────────────────────►◄
                           └─TRUE──┘
```

Parameters

TRUE
: Anonymous users can log in to an FTP server running with the FILETYPE=JES setting, and anonymous users can issue the SITE FILETYPE=JES command.

FALSE
: Anonymous users cannot log in to an FTP server running with the FILETYPE=JES setting, and anonymous users cannot issue the SITE FILETYPE=JES command.

Examples

Set the anonymous environment to allow anonymous clients to enter SITE FILETYPE=JES:

```
ANONYMOUSFILETYPEJES TRUE
```

Usage notes

If you specify the FILETYPE statement, its setting must be consistent with the ANONYMOUSFILEYTPEJES setting or anonymous users are not able to log in to FTP.

Related topics

- "ANONYMOUSFILETYPESEQ (FTP server) statement" on page 743
- "ANONYMOUSFILETYPESQL (FTP server) statement" on page 743
- "ANONYMOUSLEVEL (FTP server) statement" on page 748
- "FILETYPE (FTP client and server) statement" on page 796
- "JESGETBYDSN (FTP server) statement" on page 803

ANONYMOUSFILETYPESEQ (FTP server) statement

Use the ANONYMOUSFILETYPESEQ statement to control the access of anonymous users to the FTP server running in normal mode (FILETYPE=SEQ). This statement is recognized only when ANONYMOUSLEVEL 3 or greater is specified.

Syntax

```
    ┌─ANONYMOUSFILETYPESEQ TRUE──────┐
►►──┼────────────────────────────────┼──────────────────────────────────────►◄
    └─ANONYMOUSFILETYPESEQ──┬─TRUE──┬┘
                            └─FALSE─┘
```

Parameters

TRUE
Anonymous users can log into an FTP server running with the FILETYPE=SEQ setting, and anonymous users can issue the SITE FILETYPE=SEQ command.

FALSE
Anonymous users cannot log into an FTP server running with the FILETYPE=SEQ setting, and anonymous users cannot issue the SITE FILETYPE=SEQ command.

Examples

Set the anonymous environment to allow anonymous users to enter SITE FILETYPE=SEQ:

```
ANONYMOUSFILETYPESEQ TRUE
```

Usage notes

Most FTP servers allow anonymous users to use filetype SEQ.

If you specify the FILETYPE statement in FTP.DATA, its setting must be consistent with ANONYMOUSFILETYPESEQ or anonymous users are not able to log in to FTP.

Related topics

- "ANONYMOUSFILETYPEJES (FTP server) statement" on page 742
- "ANONYMOUSFILETYPESQL (FTP server) statement"
- "ANONYMOUSLEVEL (FTP server) statement" on page 748
- "FILETYPE (FTP client and server) statement" on page 796

ANONYMOUSFILETYPESQL (FTP server) statement

Use the ANONYMOUSFILETYPESQL statement to control the access of anonymous users to the FTP server running in SQL mode (FILETYPE=SQL).

Restriction: This statement is recognized only when ANONYMOUSLEVEL 3 or greater is specified.

Syntax

```
    ┌─ANONYMOUSFILETYPESQL FALSE───────┐
►►──┤                                  ├──────────────────────────────────────────►◄
    └─ANONYMOUSFILETYPESQL──┬─FALSE─┬──┘
                            └─TRUE──┘
```

Parameters

TRUE
Anonymous users can log into an FTP server running with the FILETYPE=SQLsetting, and anonymous users can issue the SITE FILETYPE=SQL command.

FALSE
Anonymous users cannot log into an FTP server running with the FILETYPE=SQL setting, and anonymous users cannot issue the SITE FILETYPE=SQL command.

Examples

Set the anonymous environment to allow anonymous users to enter SITE FILETYPE=SQL:

```
ANONYMOUSFILETYPESQL TRUE
```

Usage notes

If you specify the FILETYPE statement, its setting must be consistent with the ANONYMOUSFILEYTPESQL setting or anonymous users are not able to log in to FTP.

Related topics

- “ANONYMOUSFILETYPEJES (FTP server) statement” on page 742
- “ANONYMOUSFILETYPESEQ (FTP server) statement” on page 743
- “ANONYMOUSLEVEL (FTP server) statement” on page 748
- “FILETYPE (FTP client and server) statement” on page 796

ANONYMOUSFTPLOGGING (FTP server) statement

Use the ANONYMOUSFTPLOGGING statement to indicate whether the FTP server should log FTP server activity for an anonymous user. The following types of activities are logged:

- Connectivity
- Authentication
- Access
- Allocation
- Deallocation
- Data transfer
- JES job submission
- SQL query
- Abnormal end

The activities are logged in the SYSLOGD file. Each logging entry has a message number.

Syntax

```
     ┌─ANONYMOUSFTPLOGGING FALSE──────┐
►►───┤                                ├──────────────────────────────────────►◄
     └─ANONYMOUSFTPLOGGING───┬─TRUE───┘
                             └─FALSE──┘
```

Parameters

TRUE
The FTP server should log FTP session activity.

When ANONYMOUSFTPLOGGING is TRUE, a long delay in login processing might occur because the FTP server issues a DNS query to resolve the remote host IP address.

FALSE
The FTP server should not log FTP session activity.

Examples

To request that the FTP server log session activity for an anonymous user:

```
ANONYMOUSFTPLOGGING TRUE
```

Usage notes

- Each activity logging message has a message number within the range of EZYFS50 to EZYFS95.
- ANONYMOUSFTPLOGGING controls logging for anonymous users.
- If ANONYMOUSFTPLOGGING is TRUE, connectivity, authentication, and access activity log entries are made for all sessions because the server does not know whether the login is anonymous or not.

Related topics

See "FTPLOGGING (FTP server) statement" on page 797 to control logging for a non-anonymous user.

ANONYMOUSHFSDIRMODE (FTP server) statement

Use the ANONYMOUSHFSDIRMODE statement to specify the mode bits used for directories created by anonymous users.

Restriction: This statement is recognized only when ANONYMOUSLEVEL 3 or greater is specified.

Syntax

```
     ┌─ANONYMOUSHFSDIRMODE 333──┐
►►───┤                          ├────────────────────────────────────────────►◄
     └─ANONYMOUSHFSDIRMODE ─nnn─┘
```

Parameters

nnn

The three octal digits that describe the mode bits. It is passed directly to chmod() function to set the mode bits for directories created by anonymous users.

Examples

To prevent anyone from listing new directories created by anonymous users, use the following example.

```
ANONYMOUSHFSDIRMODE 333
```

Usage notes

- This statement is recognized only when ANONYMOUSFILEACCESS HFS or ANONYMOUSFILEACCESS BOTH is specified.

Related topics

- "ANONYMOUSFILEACCESS (FTP server) statement" on page 741
- "ANONYMOUSHFSFILEMODE (FTP server) statement"
- "ANONYMOUSLEVEL (FTP server) statement" on page 748

ANONYMOUSHFSFILEMODE (FTP server) statement

Use the ANONYMOUSHFSFILEMODE statement to specify the mode bits used when storing files created by anonymous users.

Restriction: This statement is recognized only when ANONYMOUSLEVEL 3 or greater is specified. This statement has no meaning if ANONYMOUSLEVEL 3 is not specified.

Syntax

```
     ┌─ANONYMOUSHFSFILEMODE 000────┐
►►───┼─────────────────────────────┼──────────────────────────────────────────►◄
     └─ANONYMOUSHFSFILEMODE ──nnn──┘
```

Parameters

nnn

The three octal digits describing the mode bits. It is passed directly to the chmod() function to set the mode bits for files created by anonymous users.

Examples

To prevent anyone from accessing files written by anonymous users, use the following example.

```
ANONYMOUSHFSFILEMODE 000
```

Usage notes

- This statement is recognized only when ANONYMOUSFILEACCESS HFS or ANONYMOUSFILEACCESS BOTH is specified.

Related topics

- "ANONYMOUSFILEACCESS (FTP server) statement" on page 741
- "ANONYMOUSHFSDIRMODE (FTP server) statement" on page 745
- "ANONYMOUSLEVEL (FTP server) statement" on page 748

ANONYMOUSHFSINFO (FTP server) statement

Use the ANONYMOUSHFSINFO statement to specify a file containing information messages specific to each FTP server directory during an FTP login session.

Restriction: This statement affects only FTP clients logged in as anonymous users.

Syntax

```
►►──ANONYMOUSHFSINFO──file-mask──────────────────────────────────────────►◄
```

Parameters

file-mask
: The file-mask is an z/OS UNIX file mask used to find a z/OS UNIX information file for anonymous users. The file mask can contain wildcards or it can be a full file name (for example, readme*). When a user changes directories, a search is made with the specified mask. The contents of the first file found is returned to the FTP client and is displayed to the end user. If no file matches the specified mask, no information is displayed to the end user. If multiple files satisfy a generic file-mask, the first is chosen.

  Restriction: The generic file name only works when an asterisk (*) is at the end of a character string.

Examples

Use the following example to display the contents of the first file matching readme* in any z/OS UNIX directory to which an anonymous user changes. If the directory has no files matching readme*, no messages are displayed.

```
ANONYMOUSHFSINFO readme*
; Anonymous HFS info file-mask
; login
```

Usage notes

- If an anonymous user changes to a directory containing no files matching the file-mask, no information is displayed to the anonymous user.

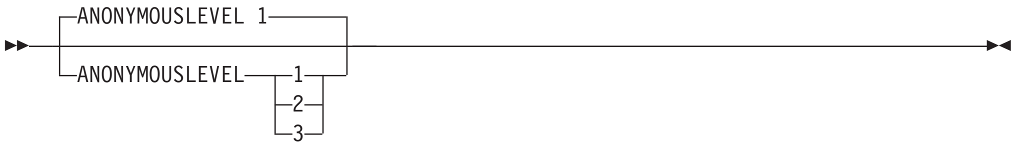
Related topics

- "ADMINEMAILADDRESS (FTP server) statement" on page 738
- "ANONYMOUS (FTP server) statement" on page 738
- "ANONYMOUSMVSINFO (FTP server) statement" on page 751
- "BANNER (FTP server) statement" on page 756
- "HFSINFO (FTP server) statement" on page 799
- "MVSINFO (FTP server) statement" on page 820

ANONYMOUSLEVEL (FTP server) statement

Use the ANONYMOUSLEVEL statement to set the type of access permitted to users who log in as anonymous users.

Syntax



Parameters

1 Anonymous logins are as documented in the ANONYMOUS statement. Anonymous users are not affected by the keywords and function of the following:

- ANONYMOUSFILETYPESEQ
- ANONYMOUSFILETYPEJES
- ANONYMOUSFILETYPESQL
- ANONYMOUSFILEACCESS
- ANONYMOUSHFSFILEMODE
- ANONYMOUSHFSDIRMODE
- EMAILADDRCHECK

2 Anonymous logins are allowed as documented in "ANONYMOUS (FTP server) statement" on page 738, except that the anonymous user's root directory is set with the UNIX call chroot() to the anonymous userid home directory. This confines the anonymous user's z/OS UNIX access to the anonymous userID home directory and its subdirectories. A umask of 777 is used for all files and directories created by anonymous users.

3 Anonymous logins are allowed as is documented in the ANONYMOUS statement, but more control is given to customize access.

The FTP.DATA statements used to give this control are:

- ANONYMOUSFILETYPESEQ
- ANONYMOUSFILETYPEJES
- ANONYMOUSFILETYPESQL
- ANONYMOUSFILEACCESS
- ANONYMOUSHFSFILEMODE
- ANONYMOUSHFSDIRMODE

The UNIX call chroot() is used to set the anonymous user's root directory to that user's home directory.

Instead of establishing a fixed UMASK for files and directories created by the anonymous user, the permission bits for files and directories are as defined by the ANONYMOUSHFSFILEMODE and ANONYMOUSHFSDIRMODE statements.

FTP clients are not allowed to issue the USER command to enter or leave anonymous login mode.

The password prompting behavior for anonymous users is different than for ANONYMOUSLEVEL 1 and 2. When the ANONYMOUS statement is coded with no user ID or password, the FTP server prompts the user to enter an e-mail address as a password. When the ANONYMOUS statement is coded with a user ID, the FTP server prompts the user to enter a password, as documented in “ANONYMOUS (FTP server) statement” on page 738. When the ANONYMOUS statement is coded with a user ID and password, the user is prompted to enter an e-mail address as a password. Control the degree of e-mail address validation with the EMAILADDRCHECK password.

When customizing FTP server to support ANONYMOUS logins, FTP server supports a way to avoid placing a plain-text password in the ANONYMOUS statement by supporting a special parameter, SURROGATE. This is shown in the following example:

```
ANONYMOUS userid/SURROGATE
```

For more information about anonymous logins, see z/OS Communications Server: IP Configuration Guide or “ANONYMOUS (FTP server) statement” on page 738.

Requirement: In order to support this function, the FTP user ID must be defined to process users without passwords.

Examples

Set the anonymous environment to use controls for accessing different resources:

```
ANONYMOUSLEVEL 3
```

Usage notes

- For ANONYMOUSLEVEL 2 and greater, when STARTDIRECTORY is z/OS UNIX, you must create a specific directory structure and contents within the anonymous user's home directory. This directory structure is needed so the FTP client maintains addressability to needed executable applications after the chroot() is executed. See z/OS Communications Server: IP Configuration Guide for details about creating the required directory structure.
- If you specify ANONYMOUSLEVEL 3 and either ANONYMOUS with no parameters or ANONYMOUS with both user ID and password, the user is prompted for an e-mail address to log in to FTP. The EMAILADDRCHECK keyword controls the extent to which the e-mail address entered is validated. See “EMAILADDRCHECK (FTP server) statement” on page 789 for more information.

Related topics

- “ANONYMOUS (FTP server) statement” on page 738
- “ANONYMOUSHFSFILEMODE (FTP server) statement” on page 746
- “ANONYMOUSHFSDIRMODE (FTP server) statement” on page 745
- “ANONYMOUSFILETYPEJES (FTP server) statement” on page 742
- “ANONYMOUSFILETYPESEQ (FTP server) statement” on page 743
- “ANONYMOUSFILETYPESQL (FTP server) statement” on page 743
- “EMAILADDRCHECK (FTP server) statement” on page 789
- “STARTDIRECTORY (FTP server) statement” on page 876

ANONYMOUSLOGINMSG (FTP server) statement

Use the ANONYMOUSLOGINMSG statement to specify a z/OS UNIX file or MVS data set whose contents are to be displayed to the end user when an anonymous user logs in.

Syntax

```
►►──ANONYMOUSLOGINMSG──file-path──────────────────────────────────────────►◄
```

Parameters

file-path
: Either a z/OS UNIX path name or a fully qualified MVS data set name. If the first character is a slash, *file-path* is considered a z/OS UNIX name; otherwise, it is treated as a fully qualified MVS data set name.

Rules:

- When specifying a z/OS UNIX file-path, *file-path* is always an absolute pathname in the anonymous user's root directory. The anonymous user's root directory depends on the values coded or set to the default value for ANONYMOUSLEVEL and STARTDIRECTORY statements in FTP.DATA.
- When ANONYMOUSLEVEL 1 is coded in FTP.DATA, or when STARTDIRECTORY MVS is coded in FTP.DATA, the anonymous user's root directory is the z/OS UNIX root directory. Therefore, you specify *file-path* as an absolute pathname in the z/OS UNIX without regard to the anonymous user's home directory.
- When the ANONYMOUSLEVEL value is greater than one, and the STARTDIRECTORY is z/OS UNIX, the anonymous user's root directory is the anonymous userID home directory. Therefore, the file identified by *file-path*has to reside within the anonymous user's home directory or one of its subdirectories, and you specify *file-path* as an absolute pathname, but relative to the anonymous userID home directory.

Examples

To display the contents of the TCPIP.ANONYM.LOGIN.MSG data set when an anonymous user logs into FTP, enter the following code:

```
ANONYMOUSLOGINMSG TCPIP.ANONYM.LOGIN.MSG
```

For example, you might have created userID GUEST with home directory /u/anonymous for anonymous logins, and you have coded these statements in FTP.DATA:

- ANONYMOUS GUEST
- ANONYMOUSLEVEL 3
- STARTDIRECTORY HFS
- ANONYMOUSFILEACCESS HFS

To display the contents of /u/anonymous/banner when an anonymous user logs into FTP, code the following statement in FTP.DATA:

```
ANONYMOUSLOGINMSG /banner
```

To display the contents of /etc/banner when an anonymous user logs into FTP, you must copy /etc/banner into /u/anonymous or into a subdirectory such as /u/anonymous/etc because the z/OS UNIX directory /etc is outside the anonymous userID's root directory.

Again, suppose you have created userID GUEST with home directory /u/anonymous for anonymous logins, and you have coded these statements in FTP.DATA:
- ANONYMOUS GUEST
- ANONYMOUSLEVEL 1

To display the contents of /u/anonymous/banner when an anymous user logs into FTP, code the following statement in FTP.DATA:

```
ANONYMOUSLOGINMSG /u/anonymous/banner
```

In this case you specify the pathname /u/anonymous/banner because the anonymous userID root directory is /.

Usage notes

- ANONYMOUSLOGINMSG is not dependent upon the value of ANONYMOUSLEVEL.
- If an installation is required to display the same login messages to both anonymous and known users, the same file-path can be specified on both the ANONYMOUSLOGINMSG and LOGINMSG statements.

Related topics

- “ANONYMOUS (FTP server) statement” on page 738
- “ANONYMOUSLEVEL (FTP server) statement” on page 748
- “ANONYMOUSFILEACCESS (FTP server) statement” on page 741
- “ANONYMOUSMVSINFO (FTP server) statement”
- “BANNER (FTP server) statement” on page 756
- “HFSINFO (FTP server) statement” on page 799
- “LOGINMSG (FTP server) statement” on page 813
- “MVSINFO (FTP server) statement” on page 820
- “STARTDIRECTORY (FTP server) statement” on page 876

ANONYMOUSMVSINFO (FTP server) statement

Use the ANONYMOUSMVSINFO statement to specify the MVS data sets whose contents should be displayed when an anonymous user changes directory. The statement identifies a low-level qualifier (LLQ) to be appended to the current path whenever an anonymous FTP user changes directories to an MVS data set.

Syntax

```
►►──ANONYMOUSMVSINFO──MVS-LLQ──────────────────────────────────────────►◄
```

Parameters

MVS-LLQ
The MVS-LLQ is the MVS low-level qualifier (LLQ) to be appended to the

current MVS path whenever an anonymous FTP user changes directories to an MVS data set. If a data set matches the current path appended LLQ, the contents of the data set are to be returned to the FTP user and displayed to the end user (when the end user is an anonymous user).

Examples

To display a readme file the first time an anonymous user changes directory to high-level qualifiers, use the statement in the following example. In this example, an MVS high-level qualifier of *productname* might have a readme file for each product, and when an anonymous user changes directory to the product, the readme file would be displayed.

```
 ANONYMOUSMVSINFO README
```

Usage notes

- You can use MVSINFO to specify the same LLQ and ANONYMOUSMVSINFO. In this way, anonymous and known users can display the same information.
- The ANONYMOUSMVSINFO data set is displayed only the first time a user changes to a specific directory. The FTP server maintains a finite history of CD commands entered by the FTP user. If the FTP user performs frequent CD commands, it is possible the user sees the same ANONYMOUSMVSINFO file more than once.
- ANONYMOUSMVSINFO applies only to anonymous users. For all other users, a banner informational message can be displayed using the MVSINFO statement.

Related topics

- "ANONYMOUS (FTP server) statement" on page 738
- "ANONYMOUSHFSINFO (FTP server) statement" on page 747
- "ANONYMOUSLOGINMSG (FTP server) statement" on page 750
- "BANNER (FTP server) statement" on page 756
- "HFSINFO (FTP server) statement" on page 799
- "MVSINFO (FTP server) statement" on page 820

|
APPLNAME (FTP server) statement

|
| Use the APPLNAME statement to specify the FTP server application name (applname).

|
Syntax

|
```
►►──APPLNAME──applname──────────────────────────────────────────────►◄
```
|

|
Parameters

| *applname*
| The FTP server application name.

|
Examples

| Use OMVSAPPL as the FTP server application name:
|
```
APPLNAME OMVSAPPL
```

Usage notes

If you do not specify any value for APPLNAME, FTP server uses job name as the application name. The maximum length of this statement is 8 bytes. Any invalid value is ignored.

ASATRANS (FTP client and server) statement

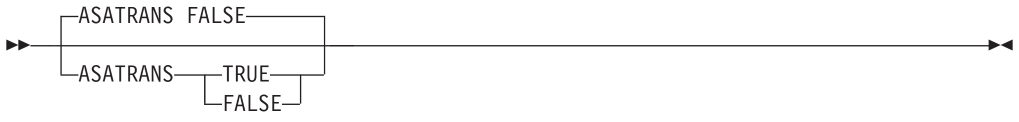
Use the ASATRANS statement to control the way ASA file transfers are managed. Choose either to have the control characters converted by the C runtime library during a file transfer or transferred without conversion.

The complete conversion process is described in the z/OS XL C/C++ Programming Guide.

Server This setting applies when transferring files from the server's system (for example, with a GET subcommand).

Client This setting applies when transferring files from the client's system (for example, with a PUT subcommand).

Syntax



Parameters

TRUE
Characters in column 1 of the file being transferred are converted to C control character sequences.

FALSE
Characters in column 1 of the file being transferred are not converted. This is the default.

Examples

Convert characters in column 1 of the file being transferred:

```
ASATRANS TRUE
```

AUTOMOUNT (FTP client and server) statement

Use the AUTOMOUNT statement to permit unmounted DASD volumes to be mounted automatically.

Server This setting applies when accessing files on the server's system.

Client This setting applies when accessing files on the client's system.

Syntax

```
      ┌─AUTOMOUNT TRUE────────┐
►►────┤                       ├──────────────────────────────────────────►◄
      └─AUTOMOUNT──┬─TRUE──┬──┘
                   └─FALSE─┘
```

Parameters

TRUE
: Permits unmounted DASD volumes to be mounted automatically. This is the default.

FALSE
: Prevents unmounted DASD volumes from being mounted automatically.

Examples

Mount DASD volumes that are not already mounted automatically:

```
AUTOMOUNT TRUE
```

Usage notes

- If AUTOMOUNT is allowed, FTP attempts to mount volumes, if necessary, to obtain temporary storage for load module transfers. Otherwise, the load module transfers fails with an `allocation failed` message if sufficient temporary storage is not already mounted and available.
- When transferring load modules, this parameter also controls whether or not the system attempts to mount additional temporary volumes if there is insufficient temporary DASD available and mounted to fulfill a load module transfer request.

AUTORECALL (FTP client and server) statement

Use the AUTORECALL statement to specify whether data sets that have been migrated by a storage manager, such as HSM, are recalled automatically.

Server This setting applies when accessing files on the server's system.

Client This setting applies when accessing files on the client's system.

Syntax

```
      ┌─AUTORECALL TRUE────────┐
►►────┤                        ├─────────────────────────────────────────►◄
      └─AUTORECALL──┬─TRUE──┬──┘
                    └─FALSE─┘
```

Parameters

TRUE
: Permits data sets migrated by the storage manager, such as HSM, to be recalled automatically. This is the default.

FALSE
: Prevents migrated data sets from being recalled automatically.

Examples

Recall migrated HSM files automatically:

```
AUTORECALL TRUE
```

Usage notes

- Migrated data sets can still be deleted even though you specify `FALSE`.
- Partitioned data set members require the entire data set to be recalled.

AUTOTAPEMOUNT (FTP client and server) statement

Use the AUTOTAPEMOUNT statement to specify whether unmounted tapes are to be automatically allocated and mounted.

Server This setting applies when accessing files on the server's system.

Client This setting applies when accessing files on the client's system.

Syntax

Server syntax

```
   ┌─AUTOTAPEMOUNT TRUE──────┐
►►─┼─────────────────────────┼──────────────────────────────────────────►◄
   └─AUTOTAPEMOUNT──┬─TRUE──┬┘
                    └─FALSE─┘
```

Client syntax

```
   ┌─AUTOTAPEMOUNT FALSE─────┐
►►─┼─────────────────────────┼──────────────────────────────────────────►◄
   └─AUTOTAPEMOUNT──┬─TRUE──┬┘
                    └─FALSE─┘
```

Parameters

`TRUE`
: Permits unmounted tapes to be automatically allocated and mounted. This is the default for the server.

`FALSE`
: Prevents unmounted tapes from being automatically allocated and mounted. This is the default for the client.

Examples

Automatically mount tape volumes that are not already mounted:

```
AUTOTAPEMOUNT TRUE
```

Do not automatically mount tape volumes that are not already mounted:

```
AUTOTAPEMOUNT FALSE
```

BANNER (FTP server) statement

Use the BANNER statement to identify the welcome banner to be displayed immediately after a client connects to the server.

Syntax

```
►►──BANNER──file-path──────────────────────────────────────────────►◄
```

Parameters

file-path
: The file path is the z/OS UNIX absolute pathname or the fully qualified MVS data set name whose contents are displayed whenever a user connects to FTP. A z/OS UNIX pathname must begin with a slash (/) character. An MVS data set must not begin with a slash character.

Examples

To display the contents /etc/ftp.banner each time an FTP client connects to the FTP server, enter the following code in the server's FTP.DATA:

```
BANNER /etc/ftp.banner ; banner to be displayed for FTP
```

Usage notes

- If no BANNER statement is specified, no banner is displayed immediately after a new connection is established.
- One hundred lines of the file are displayed to the FTP client as 220 replies. If the file exceeds 100 lines, a final 220 reply is returned to the client indicating the banner was truncated.

Related topics

- “ADMINEMAILADDRESS (FTP server) statement” on page 738
- “ANONYMOUSHFSINFO (FTP server) statement” on page 747
- “ANONYMOUSLOGINMSG (FTP server) statement” on page 750
- “ANONYMOUSMVSINFO (FTP server) statement” on page 751
- “HFSINFO (FTP server) statement” on page 799
- “MVSINFO (FTP server) statement” on page 820

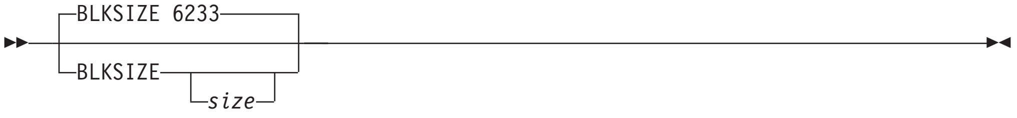
BLKSIZE (FTP client and server) statement

Use the BLKSIZE statement to specify the block size of newly allocated data sets.

Server This setting applies when creating files on the server's system (for example, with a PUT subcommand).

Client This setting applies when creating files on the client's system (for example, with a GET subcommand).

Syntax



Parameters

size
: Specifies the block size of newly allocated data sets. The valid range is 0 - 32 760. Specifying no value, or a value of 0 for block size, allows the block size from a model DCB data set or SMS dataclass to be used. The default block size is 6 233.

Examples

Set block size to 6 144 bytes:

```
BLKSIZE 6144
```

Allow the block size from a model DCB data set or SMS dataclass to be used:

```
BLKSIZE
```

Usage notes

- If you specify the BLKSIZE statement without a *size*, FTP does not specify the block size when allocating new data sets.
- The block size attribute can be obtained from an SMS data class using the DATACLASS configuration statement, from a model data set using the DCBDSN configuration statement, or from the BLKSIZE statement.
- Use BLKSIZE without a size if you have:
  - Specified a DATACLASS statement and want to use the blocksize from the data class, or
  - Specified a DCBDSN statement and want to use the blocksize from the model data set.
- If you specify a DATACLASS, a DCBDSN, and BLKSIZE without size, the value from the model data set is used.
- To override the blocksize attribute from the DATACLASS or DCBDSN settings:
  - Specify BLKSIZE with a value other than 0, or
  - Do not specify the BLKSIZE statement, and use the default.

Related topics

- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "DATACLASS (FTP client and server) statement" on page 770
- "DCBDSN (FTP client and server) statement" on page 776
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- "MGMTCLASS (FTP client and server) statement" on page 818
- "STORCLASS (FTP client and server) statement" on page 877

BUFNO (FTP client and server) statement

Use the BUFNO statement to specify the number of access method buffers used when data is read from or written to a data set.

Server This setting applies when reading or writing files on the server's system.

Client This setting applies when reading or writing files on the client's system.

Syntax

```
     ┌─BUFNO 5──────┐
►►───┼──────────────┼──────────────────────────────────────────────────►◄
     └─BUFNO number─┘
```

Parameters

number
Specifies the number of buffers allocated. The valid range is 1 - 35. The default is 5.

CCONNTIME (FTP client) statement

Use the CCONNTIME statement to specify the amount of time that the FTP client waits after attempting to close a control connection before terminating it and reporting an error.

Syntax

```
     ┌─CCONNTIME 30──────────┐
►►───┼───────────────────────┼─────────────────────────────────────────►◄
     └─CCONNTIME ──seconds───┘
```

Parameters

seconds
The number of seconds to which the timer is set. The valid range is 0 (CCONNTIME not used) or 15-86400. The default is 30 seconds.

Examples

```
CCONNTIME  60  ; wait 60 seconds
```

Related topics

See the FTP command information in the z/OS Communications Server: IP User's Guide and Commands for a description of the timeout parameter that can be used to change the timer when FTP is started.

CCTRANS (FTP client) statement

Use the CCTRANS statement to specify the SBCS translation table the FTP client uses for the control connection. The FTP client uses the translation table in the *user_id.dsn_qual*.TCPXLBIN data set. If that data set does not exist, the FTP client uses the *hlq.dsn_qua*l.TCPXLBIN data set.

Syntax

```
►►── CCTRANS ──dsn_qual──────────────────────────────────────────────────────►◄
```

Parameters

dsn_qual
The data set name qualifier for the translation table.

Examples

```
CCTRANS  CTRL  ;  use USER33.CTRL.TCPXLBIN when ftp
               ;  is used by USER33
```

Usage notes

- CTRLCONN and CCTRANS are mutually exclusive statements. If both statements appear in the FTP.DATA file, CCTRANS is ignored.
- EXTENSION UTF8 and CCTRANS are mutually exclusive statements. If both statements appear in the FTP.DATA file, CCTRANS is ignored.

Related topics

- "CTRLCONN (FTP client and server) statement" on page 769
- "EXTENSIONS (FTP client and server) statement" on page 792

CCXLATE (FTP server) statement

Use the CCXLATE statement to specify a data set containing translate tables to be used for the control connection.

Syntax

```
►►──CCXLATE──name────────────────────────────────────────────────────────────►◄
```

Parameters

name
Specifies a 1- to 8-character name corresponding to a data set containing translate tables.

FTP looks first for an environment variable called _FTPXLATE_*name*. If the environment variable exists, its value is used as the data set name

Restriction: The environment variable name must be all uppercase, although the CCXLATE parameter can be in mixed case.

If the environment variable does not exist, FTP looks for a data set called *hlq.name*.TCPXLBIN.

Examples

```
CCXLATE FRED
```

If environment variable _FTPXLATE_FRED=FREDDYS.TABLES is defined for the FTP server, this statement specifies that the translate tables in data set FREDDYS.TABLES should be used for the control connection.

If there is no such environment variable defined, this statement specifies that the translate tables data set *hlq*.FRED.TCPXLBIN should be used.

Usage notes

- CCXLATE and CTRLCONN are mutually exclusive statements. If both statements appear in your FTP.DATA file, CCXLATE is ignored.
- The CCXLATE statement (and its value) is not case sensitive but the name of the corresponding environment variable must be all uppercase or FTP does not recognize it.
- CCXLATE and EXTENSIONS UTF8 are mutually exclusive statements. If both statements appear in FTP.DATA, the CCXLATE statement is ignored.

Related topics

- Appendix A, "Translation tables," on page 1417
- "CTRLCONN (FTP client and server) statement" on page 769
- "EXTENSIONS (FTP client and server) statement" on page 792
- See z/OS Communications Server: IP Configuration Guide for more information about defining optional environment variables.
- To see the search order that determines the conversion for the control connection, see "SBCS translation table hierarchy" on page 1418.
- "XLATE (FTP server) statement" on page 897

CHKCONFIDENCE statement (FTP client and server) statement

Use the CHKCONFIDENCE statement to tell the FTP client or server whether to check and report on the confidence level in the successful completion of file transfers. Checks include reporting a missing EOF marker in an inbound data set being transferred using record structure (STRUCTURE RECORD), or block mode (MODE B), or compress mode (MODE C) and verifying that the sender is still responding after the transfer.

Server The server reports the confidence level after each transfer with FTP log message EZYFS86I and with a parameter passed to the FTPOSTPR user exit.

Client The client reports the confidence level after each file transfer by issuing message EZA2108I.

Tips:

- If the MBREQUIRELASTEOL statement is set to FALSE, the confidence level reported when a multibyte file is received from the network without an EOL sequence in the last record is `High`.
- If the MBREQUIRELASTEOL statement is set to TRUE, the confidence level reported when a multibyte file is received from the network without an EOL sequence in the last record is `Low`.

Syntax

```
   ┌─CHKCONFIDENCE FALSE──────┐
►►─┼──────────────────────────┼──────────────────────────────────────────────►◄
   └─CHKCONFIDENCE──┬─TRUE──┬─┘
                    └─FALSE─┘
```

Parameters

TRUE

Perform the following checks and report any detected conditions that cast doubt on the successful completion of a transfer.

Use the following to make the determination:

- Whether a missing EOF marker condition is detected in an inbound STRUCTURE RECORD, or MODE B, or MODE C file
- Whether the sender fails to respond following any type of transfer
- Whether some other condition causes the transfer to fail or establish doubt about its completion

FALSE

Do not perform the checks or report on the confidence level in the successful completion of a transfer. This does not suppress reporting of error conditions.

Tips: Consider the following information when using the CHKCONFIDENCE statement:

- A missing EOF marker might or might not signal an error in the transmission, and it is reported only if no other problem is detected. A confidence level of NoEOF reflects a missing EOF marker. Any other problem changes the confidence level to Low.
- **FTP client**
  - See the message information for EZA2108I in z/OS Communications Server: IP Messages Volume 1 (EZA) for more information.
- **FTP server**
  - Either code FTPLOGGING TRUE in FTP.DATA or install the FTPOSTPR exit routine, to determine what confidence level the server assigns to each file transfer.
  - Message EZYFS86I is logged only when FTPLOGGING TRUE is coded in FTP.DATA. For additional information about EZYFS86I, see z/OS Communications Server: IP Messages Volume 3 (EZY).
  - The confidence level is passed to the FTPOSTPR exit. See “The FTPOSTPR user exit” on page 703 for information about the FTPOSTPR exit routine.

Related topics

- “FTPLOGGING (FTP server) statement” on page 797
- “The FTPOSTPR user exit” on page 703

CHKPTFLUSH (FTP client) statement

If FTP saves checkpoint information in the checkpoint file or data set, z/OS might buffer the records in volatile storage instead of writing the data to storage media immediately. Use the CHKPTFLUSH statement to specify whether FTP forces z/OS to flush checkpoint information to storage media when each record is written, or allows z/OS to determine when checkpoints are flushed from volatile storage to storage media.

Guideline: When you allow z/OS to buffer checkpoint records, if you configure a large checkpoint interval and your operator cancels a file transfer operation when check pointing is active, you might lose most or all of the checkpoint data. This is because z/OS cannot flush buffered data to storage media when the FTP job is canceled. If you lose most or all of the checkpoint data, you cannot restart the file transfer from the point where it is interrupted. This is inefficient for long running file transfer operations. Code CHKPTFLUSH TRUE if this is a problem at your installation.

Syntax

```
     ┌─CHKPTFLUSH FALSE───────┐
►►───┼────────────────────────┼────────────────────────────────────────────────►◄
     └─CHKPTFLUSH──┬─FALSE─┬──┘
                   └─TRUE──┘
```

Parameters

FALSE
z/OS is allowed to buffer checkpoint records in volatile storage when FTP saves them. z/OS determines when to flush buffered records to storage media. This is the default value.

TRUE
z/OS flushes checkpoint records to storage media as soon as FTP saves them.

Examples

z/OS flushes checkpoint records to storage media as soon as FTP saves them.

```
CHKPTFLUSH TRUE
```

Usage notes

None

Related topics

- "CHKPTINT (FTP client and server) statement"
- "CHKPTPREFIX (FTP client) statement" on page 764

CHKPTINT (FTP client and server) statement

Use the CHKPTINT statement to specify the number of records that can be sent between restart markers when transferring files in EBCDIC block mode or EBCDIC compress mode when the file type is SEQ.

Server This setting applies when the server is the sending site (when the server is processing the RETR command).

The server ignores this setting when file type is not SEQ.

The server ignores this setting when it is retrieving data from a z/OS UNIX named pipe.

Requirement: Do not specify a nonzero value unless the client supports the checkpoint and restart function.

Client This setting applies when the client is processing the APPEnd, PUt, and MPUt subcommands. When you have configured RESTGET TRUE at the FTP client, this setting applies also to the GEt and MGEt subcommands. For more information about configuring RESTGET, see the locsite subcommand in the z/OS Communications Server: IP User's Guide and Commands and “RESTGET (FTP client) statement” on page 837.

The client ignores this setting when file type is not SEQ.

The client ignores this setting when you are transferring data to or from a z/OS UNIX named pipe.

Rule: Do not specify a nonzero value unless the FTP server supports the RESTart command and checkpoint and restart function.

Syntax

```
   ┌─CHKPTINT 0──────────┐
►►─┼─────────────────────┼──────────────────────────────────────────────────►◄
   └─CHKPTINT──number────┘
```

Parameters

number
Used to determine when a restart marker is transmitted. The marker is transmitted after the specified number of records are sent.

If the *number* value is set to 0, checkpointing does not occur and no marker blocks are transmitted. The default is 0.

Examples

To send a restart marker of every 100000 records when the client is the sending site:

Client's FTP.DATA:

```
CHKPTINT 100000
```

To enable the checkpoint restart function when the server is the sending site, code the following statements in FTP.DATA:

Client's FTP.DATA:

```
RESTGET  TRUE
```

Server's FTP.DATA:

```
CHKPTINT any non-zero value
```

Usage notes

- Specify a nonzero value to enable checkpointing during a file transfer. When checkpointing is enabled during a file transfer, you can restart a failed file transfer. To restart a failed transfer from the z/OS FTP client, use the restart subcommand. See the restart subcommand information in z/OS Communications Server: IP User's Guide and Commands.
- Client and server must both support the checkpoint/restart function. From the z/OS FTP client, you can enable or disable the client's support after logging in with a locsite subcommand. See z/OS Communications Server: IP User's Guide and Commands for more information.
- The z/OS FTP server allows you to change the value with a SITE CHKPTINT command. If only certain clients support the restart function, you should code CHKPTINT 0 in the server's FTP.DATA and direct the user to use a SITE

command to set the server value after logging in. See the SITE command information in z/OS Communications Server: IP User's Guide and Commands for more information.

Related topics

- "CONDDISP (FTP client and server) statement" on page 768
- "CHKPTPREFIX (FTP client) statement"
- "RESTGET (FTP client) statement" on page 837
- "RESTPUT (FTP server) statement" on page 838

CHKPTPREFIX (FTP client) statement

Use the CHKPTPREFIX statement to specify the high level qualifier (*hlq*) for the FTP client checkpoint file. The FTP client uses the *hlq* to determine the name of the local checkpoint data set or file.

Syntax

```
    ┌─CHKPTPREFIX HOME────────┐
►►──┼─────────────────────────┼──────────────────────────────────────────►◄
    └─CHKPTPREFIX───┬─HOME───┬┘
                    ├─USERID─┤
                    └─LOCAL──┘
```

Parameters

HOME
: If the client is running in the z/OS UNIX shell, the *hlq* is the current path and the name of the checkpoint file is *current_path*/ftp.chkpoint. Otherwise, the *hlq* is the TSO prefix and the checkpoint data set is named *tso_prefix*.FTP.CHKPOINT.

  This is the default.

USERID
: Use the user ID associated with the address space where the FTP command is issued as the *hlq* for the checkpoint data set. The name of the data set is userID.FTP.CHKPOINT.

LOCAL
: Use the local working directory (local_dir) as set by the lcd subcommand. If the directory is a z/OS UNIX directory, the checkpoint file is local_dir/ftp.chkpoint. If the directory is a partitioned data set, the checkpoint data set name is local_dir(CHKPOINT). Otherwise, the checkpoint data set is local_dir.FTP.CHKPOIN'.

Examples

To use the user ID, use the following code:

```
CHKPTPREFIX  USERID
```

Usage notes

None

Related topics

- "CHKPTINT (FTP client and server) statement" on page 762
- "RESTGET (FTP client) statement" on page 837

CIPHERSUITE (FTP client and server) statement

Use the CIPHERSUITE statement to specify the name of a cipher algorithm that is used during the TLS handshake.

Server Indicates the server's preference of cipher algorithms.

Client Indicates the client's preference of cipher algorithms.

Syntax

```
►►──CIPHERSUITE──name──────────────────────────────────────────────►◄
```

Parameters

name

The name of the cipher algorithm. The following values are allowed *name* values:

- SSL_NULL_MD5
- SSL_NULL_SHA
- SSL_RC4_MD5_EX
- SSL_RC4_MD5
- SSL_RC4_SHA
- SSL_RC2_MD5_EX
- SSL_DES_SHA
- SSL_3DES_SHA
- SSL_AES_128_SHA
- SSL_AES_256_SHA

The *name* can be interpreted as follows:

`SSL_<cipher>_<cipher hash>[_EX]`

<*cipher*> specifies one of the following encryption algorithms:

AES_128
128–bit AES; Advanced Encryption Standard is established by the National Institute of Standards and Technology (NIST).

AES_256
256–bit AES; Advanced Encryption Standard is established by the National Institute of Standards and Technology (NIST).

RC2 Block cipher developed at RSA Data Security

RC4 Stream cipher developed at RSA Data Security

DES Digital Encryption Standard (56 bits of security)

3DES Digital Encryption Standard (168 bits of security)

NULL No algorithm is used. NULL indicates that there is no key exchange.

<*cipher hash*> specifies one of the following authentication algorithms:

MD5	Algorithm that converts to fixed size (16 bytes)
SHA	Secure Hash Algorithm that converts to a 20-byte output

The suffix _EX indicates that the corresponding cipher suite is exportable.

Restrictions:

- The following list shows the subject to export restrictions and might not be available outside of the United States:
  - SSL_3DES_SHA
  - SSL_RC4_SHA
  - SSL_RC4_MD5
  - SSL_AES_128_SHA
  - SSL_AES_256_SHA
- Only RSA key exchange is supported.

Examples

To indicate that you want to use the 3DES encryption and SHA authentication as your first choice, and that RC4 encryption and MD5 authentication are your second choice, code the following examples:

```
CIPHERSUITE SSL_3DES_SHA
CIPHERSUITE SSL_RC4_MD5
```

Authorization

- Multiple CIPHERSUITE statements can be coded in the FTP.DATA file.
- The order of CIPHERSUITE statements in the server's FTP.DATA file indicates the priority of the algorithms listed. Specify the highest priority algorithm first in the FTP.DATA file.
- The client and server specify the list of encryption types that they support. The client and server negotiate which of the available ciphers is used for the data encryption by specifying the desired ciphers in order of preference. The actual cipher used is the best match between what the server supports and what the client requests. If the server does not support any of the ciphers that the client requests, the TLS handshake fails and the connection is closed. See the z/OS Cryptographic Services System SSL Programming for a list of ciphers that are included in the base product.
- The CIPHERSUITE statements are used by the FTP server when the EXTENSIONS statement is coded with the AUTH_TLS value.
- The CIPHERSUITE statements are used by the FTP client when the SECURE_MECHANISM TLS statement is coded or when the FTP client is started with either the -a TLS or the -r TLS start parameter.

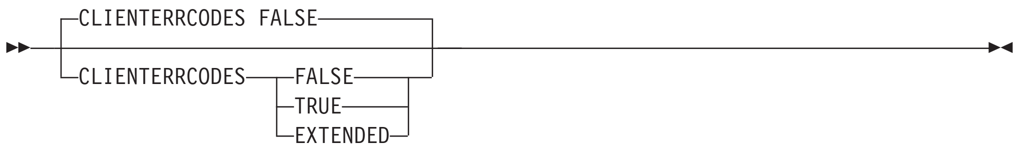
Related topics

- “EXTENSIONS (FTP client and server) statement” on page 792
- “SECURE_MECHANISM (FTP client) statement” on page 854
- “TLSMECHANISM (FTP client and server) statement” on page 879
- See z/OS Communications Server: IP Configuration Guide for more information about customizing TLS and Kerberos.

CLIENTERRCODES (FTP client) statement

Use the CLIENTERRCODES (FTP client) to specify whether FTP return codes are to be converted to client error codes.

Syntax



Parameters

FALSE
Issue standard FTP return codes. See the FTP return codes topic in z/OS Communications Server: IP User's Guide and Commands for a complete description of standard FTP return codes.

TRUE
Convert FTP return codes into a set of codes defined in FTP client error codes in the z/OS Communications Server: IP User's Guide and Commands.

EXTENDED
Convert FTP return codes into the client error code (as would be returned for TRUE) concatenated with the subcommand number. See the FTP client error codes extended topic of the z/OS Communications Server: IP User's Guide and Commands for more information.

Examples

```
CLIENTERRCODES  EXTENDED ; request extended error codes
```

Usage notes

When the FTP client is invoked from the FTP client application programming Interface (API), the value on the CLIENTERRCODES statement does not affect the operation of the client, as all return codes including client error codes are returned to the application.

Related topics

- “LOGCLIENTERR (FTP client) statement” on page 812
- FTP client error logging in z/OS Communications Server: IP User's Guide and Commands
- For more information, see using the FTP client API trace in z/OS Communications Server: IP Programmer's Guide and Reference.

CLIENTEXIT (FTP client) statement

Use the CLIENTEXIT statement to specify whether the FTP client exits with a nonzero MVS return code for certain FTP errors.

Syntax

```
   ┌─CLIENTEXIT FALSE─────┐
►►─┤                      ├─────────────────────────────────────────────────►◄
   └─CLIENTEXIT──┬─FALSE─┬┘
                 └─TRUE──┘
```

Parameters

FALSE
FTP client does not exit with a nonzero MVS return code when certain errors occur.

Result: Starting the FTP client with the EXIT parameter can override CLIENTEXIT FALSE.

TRUE
FTP client exits with a nonzero MVS return code when certain errors occur.

Result: Coding CLIENTEXIT TRUE is equivalent to starting the FTP client with the EXIT parameter. See FTP command — Entering the FTP environment in the z/OS Communications Server: IP User's Guide and Commands for more information about the EXIT parameter.

Examples

```
CLIENTEXIT TRUE ; FTP client will exit when an error occurs
                ; even though EXIT is not coded  specified when
                ; starting the FTP client.
```

Related topics

- "CLIENTERRCODES (FTP client) statement" on page 767
- FTP command — Entering the FTP environment in the z/OS Communications Server: IP User's Guide and Commands
- FTP return codes in the z/OS Communications Server: IP User's Guide and Commands

CONDDISP (FTP client and server) statement

Specify whether to keep or delete a new data set, z/OS UNIX file, or z/OS UNIX named pipe when an FTP file transfer ends prematurely.

Server This setting applies when writing new files, named pipes, or data sets on the server system (for example, with a PUT subcommand).

Client This setting applies when writing new files, named pipes, or data sets on the client system (for example, with a GET subcommand).

Syntax

```
   ┌─CONDDISP CATLG──────┐
►►─┤                     ├──────────────────────────────────────────────────►◄
   └─CONDDISP──┬─CATLG──┬┘
               └─DELETE─┘
```

Parameters

CATLG
Specifies that new data sets, z/OS UNIX files, and z/OS UNIX named pipes are kept when an FTP file transfer ends prematurely. For MVS data set transfers, the data set is also cataloged. This is the default.

DELETE
Specifies that new data sets, z/OS UNIX files, and z/OS UNIX named pipes are deleted when a file transfer ends prematurely.

Examples

Specify that a new data set, z/OS UNIX file, or named pipe is deleted when a file transfer ends prematurely:

```
CONDDISP DELETE
```

Rules:

- DELETE is ignored if the file transfer ended prematurely because FTP was stopped.
- DELETE is ignored if a checkpoint marker is received.
- If you are running a job scheduling program that detects files as they are cataloged and then schedules a subsequent job for processing, the job scheduler must take into account that setting CONDDISP=DELETE causes FTP to delete and uncatalog the data set when the file transfer fails. For generation data groups, the following situations might occur:
  - FTP intends to create a new GDG(+1) and generates GDG.G00023V00.
  - The transfer of this data set fails, and the GDG.G00023V00 data set is deleted and uncataloged.
  - A follow-on reference for the current GDG, for example, GDG(0), would cause the data set GDG.G00022V00 to be accessed and old data to be processed.
- If you are transferring a physical sequential data set with the MVSGet or MVSPut subcommand, the data set that is created is disposed according to the CONDDISP configuration if the transfer ends prematurely. However, if you are transferring a PDS or library data set, the data set that is created is deleted regardless of the CONDDISP configuration if the transfer ends prematurely.

Related topic

- "CHKPTINT (FTP client and server) statement" on page 762

CTRLCONN (FTP client and server) statement

This statement defines the ASCII code page to be used for the control connection.

Server Specifies the code page used by the server.

Client Specifies the code page used by the client.

Syntax

```
      ┌─CTRLCONN 7BIT──────────────────────┐
►►────┼────────────────────────────────────┼──────────────────────────────────────────────►◄
      └─CTRLCONN──┬─7BIT──────────────────┬┘
                  ├─iconv_ascii───────────┤
                  └─FTP_STANDARD_TABLE────┘
```

Parameters

7BIT
Indicates that the 7-bit ASCII code page is to be used. 7BIT is the default if CTRLCONN is not used and no TCPXLBIN data set is found.

iconv_ascii
A name recognized by iconv to indicate an ASCII code page.

FTP_STANDARD_TABLE
Indicates that the FTP internal tables, which are the same as the tables that are shipped in TCPXLBIN(STANDARD), are to be used.

Examples

```
CTRLCONN IBM-858
```

Usage notes

- 7BIT or an *iconv_ascii* name can be entered in lowercase or uppercase.
- To see the search order that determines the code page conversion for the control connection, see z/OS Communications Server: IP Configuration Guide.
- EXTENSIONS UTF8 and CTRLCONN are mutually exclusive statements. If both statements are coded in FTP.DATA, CTRLCONN is ignored.

Related topics

- "CCXLATE (FTP server) statement" on page 759
- "EXTENSIONS (FTP client and server) statement" on page 792
- For the code pages supported, see code set converters in the z/OS XL C/C++ Programming Guide.

DATACLASS (FTP client and server) statement

Use the DATACLASS statement to specify the SMS-managed data class as defined by your organization for the FTP server.

Server This setting applies when creating files on the server's system (for example, with a PUT subcommand).

Client This setting applies when creating files on the client's system (for example, with a GET subcommand).

Syntax

```
►►──DATACLASS──class──────────────────────────────────────────────────────────────►◄
```

Parameters

class
: The SMS-managed data class as defined by your organization. There is no default.

Examples

Use the SMS data class SMSDATA when allocating new data sets:

```
DATACLASS SMSDATA
```

Results:

- If you code any of the following FTP.DATA statements or let FTP assign them default values, the configured values or default values override the values specified in the SMS DATACLASS:
  - BLKSIZE
  - DIRECTORY
  - LRECL
  - PRIMARY
  - RECFM
  - RETPD
  - SECONDARY

  Guideline: To prevent these statements from overriding the values specified in the SMS DATACLASS, perform one of the following actions:
  - Code the statements with no parameters.
  - For client allocations, use the LOCSIte subcommand to configure these options with no values before allocating a new data set.
  - For server allocations, use the SIte subcommand or the server SITE command to configure these options with no values before allocating a new data set.
- If you code the following statements in FTP.DATA with the default value or let FTP assign the default value, FTP will use the value in the SMS DATACLASS.
  - DNSTYPE
  - EATTR

  If the SMS DATACLASS does not specify the value, FTP will use the system default.

  Guideline: To override the SMS DATACLASS value and the system default value, perform one of the following actions:
  - Code statements in FTP.DATA with the values that you want.
  - For client allocations, use the LOCSIte subcommand to configure the values that you want.
  - For server allocations, use the SIte subcommand or the server SITE command to configure the values that you want.
- The PDSTYPE FTP.DATA statement has no default values. If you code this statement in FTP.DATA with parameters, the configured value overrides the value specified in the SMS DATACLASS.

  Guideline: To prevent this statement from overriding the value specified in the SMS DATACLASS, perform one of the following actions:
  - Code the statement with no parameter.
  - Remove the statement from FTP.DATA.

- For client allocations, use the LOCSIte subcommand to configure this option with no value before allocating a new data set.
- For server allocations, use the SIte subcommand or the server SITE command to configure this option with no value before allocating a new data set.

- If you specify the DCBDSN statement, the LRECL, RECFM, BLKSIZE, and RETPD (if specified) of the DCBDSN data set override the values specified in the SMS DATACLASS.

  Guideline: To prevent the DCBDSN values from overriding the values specified in the SMS DATACLASS, perform one of the following actions:
  - Code statements in FTP.DATA for LRECL, RECFM, BLKSIZE, and RETPD with no keyword values.
  - For client allocations, use the LOCSIte subcommand to configure these options with no values before allocating a new data set.
  - For server allocations, use the SIte subcommand or the server SITE command to configure these options with no values before allocating a new data set.
- If you specify the MGMTCLASS statement and the requested management class specifies a retention period, the RETPD value of the management class might override the RETPD value of DATACLASS.

Related topics:
- “BLKSIZE (FTP client and server) statement” on page 756
- “DCBDSN (FTP client and server) statement” on page 776
- “DIRECTORY (FTP client and server) statement” on page 781
- “DSNTYPE (FTP client and server) statement” on page 783
- “EATTR (FTP client and server) statement” on page 788
- “LRECL (FTP client and server) statement” on page 813
- “MGMTCLASS (FTP client and server) statement” on page 818
- “PDSTYPE (FTP client and server) statement” on page 826
- “PRIMARY (FTP client and server) statement” on page 830
- “RECFM (FTP client and server) statement” on page 833
- “RETPD (FTP client and server) statement” on page 838
- “SECONDARY (FTP client and server) statement” on page 845
- “STORCLASS (FTP client and server) statement” on page 877
- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.

DATACTTIME (FTP client) statement

Use the DATACTTIME statement to specify the number of seconds that the FTP client waits after attempting to send or receive data before terminating the connection and reporting an error to the user. The default is 120. The valid range for DATACTTIME is 0 (DATACTTIME not used) or 15-86 400.

Syntax

```
   ┌─DATACTTIME 120──────┐
►►─┼─────────────────────┼──────────────────────────────────────────►◄
   └─DATACTTIME ──seconds─┘
```

Parameters

seconds
: The number of seconds to which the timer is set. The valid range is 0 (DATACTTIME not used) or 15-86 400. The default is 120 seconds.

Examples

```
DATACTTIME  160  ; wait 160 seconds
```

Related topics

See the FTP command and the FTP environment information in z/OS Communications Server: IP User's Guide and Commands.

DATAKEEPALIVE (FTP client and server) statement

Use the DATAKEEPALIVE statement to specify the data connection keepalive timer.

Results:

- The DATAKEEPALIVE statement overrides the keepalive timer value that you configured in the PROFILE.TCPIP file.
- The keepalive timer causes TCP/IP to send a keepalive packet on the data connection when the connection is idle for the length of time specified in the DATAKEEPALIVE statement. Keepalive packets prevent the data connection from timing out as a result of long periods of inactivity.

Server Specifies how often the server sends a keepalive packet.

Client Specifies how often the client sends a keepalive packet.

Syntax

```
    ┌─DATAKEEPALIVE 0──────────┐
►►──┼──────────────────────────┼──────────────────────────────────────►◄
    └─DATAKEEPALIVE ──seconds──┘
```

Parameters

seconds
: The number of seconds of inactivity that passes before a keepalive packet is sent out on the FTP data connection. Valid values are 0 (DATAKEEPALIVE not used) or 60 - 86 400. The default is 0.

Rule: If you specify 0 seconds, the DATAKEEPALIVE timer is disabled, and the only keepalive packets that flow on the data connection are controlled by the interval for the keepalive packets that you configured in the stack.

Guidelines:

- Use the DATAKEEPALIVE statement if the DSWAITTIME configuration option is a value other than 0.
- Use the DATAKEEPALIVE statement for FILETYPE=JES transfers.

Examples

Use the following code to set the data connection keepalive timer to 60 seconds:

```
DATAKEEPALIVE 60
```

Related topics

- "DSWAITTIME (FTP client and server) statement" on page 784
- "FTPKEEPALIVE (FTP client and server) statement" on page 797

DATATIMEOUT (FTP server) statement

Use the DATATIMEOUT statement to specify the length of time to wait for the send to complete before the connection is aborted.

Syntax

```
      ┌─DATATIMEOUT 300──────┐
►►────┼──────────────────────┼──────────────────────────────────────────────►◄
      └─DATATIMEOUT──seconds─┘
```

Parameters

seconds
: Used to determine when to abort the connection if a send() or recv() is not completed or if a passive socket was opened, but never completed by the remote client. Allowed values are 0 - 86 400. The default is 300 seconds.

  Specifying 0 indicates no timeout value is used, and the transfer does not time out.

Examples

To check for send completion at 30 seconds:

```
DATATIMEOUT 30
```

Usage notes

The DATATIMEOUT timer is set when the FTP server does a send() or recv() call to TCP/IP or when a passive data connection is detected, and the server must wait for the client to complete the session. If the process does not complete within the timer value, the connection is aborted.

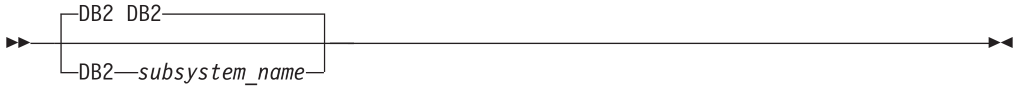
DB2 (FTP client and server) statement

Use the DB2 statement to specify the name of the DB2 subsystem.

Server This setting applies when FILETYPE=SQL and a GET subcommand is processed.

Client This setting applies when FILETYPE=SQL and a PUT subcommand is processed.

Syntax



Parameters

subsystem_name
The name of the DB2 subsystem. The default name is DB2.

Examples

Set the DB2 subsystem name to DB2X:

```
DB2 DB2X
```

Related topics

- "DB2PLAN (FTP cilent and server) statement"
- See the DB2 SQL queries information in z/OS Communications Server: IP User's Guide and Commands.
- See the information about the SQL query function in z/OS Communications Server: IP Configuration Guide.

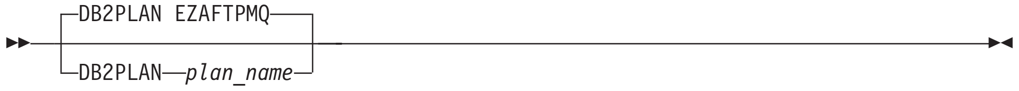
DB2PLAN (FTP cilent and server) statement

Use the DB2PLAN statement to specify the DB2 plan to be used by the FTP server.

Server This setting applies when FILETYPE=SQL and a GET subcommand is processed.

Client This setting applies when FILETYPE=SQL and a PUT subcommand is processed.

Syntax



Parameters

plan_name
The name of the DB2 plan bound in the DB2 subsystem.

Examples

Set the plan name to FTPPLAN:

```
DB2PLAN   FTPPLAN
```

Related topic

- "DB2 (FTP client and server) statement" on page 774

DBSUB (FTP client and server) statement

Use the DBSUB statement in server and client FTP.DATA to specify whether substitution is allowed for double-byte data that cannot be translated. The site and locsite subcommands are also available to set this keyword.

Server Specifies whether double-byte substitution is allowed on the server's system.

Client Specifies whether double-byte substitution is allowed on the client's system.

Syntax

```
   ┌─DBSUB FALSE──────────┐
►►─┤                      ├──────────────────────────────────────────────►◄
   └─DBSUB───┬─TRUE──┬────┘
             └─FALSE─┘
```

Parameters

FALSE
Substitution is not allowed for double-byte character translation. This causes a data transfer failure if a character cannot be mapped during the transfer. This is the default value.

TRUE
Substitution is allowed for double-byte character translation.

Examples

To allow substitution for double-byte character translation, use the following code:

```
DBSUB TRUE
```

DCBDSN (FTP client and server) statement

Use the DCBDSN statement to specify an MVS data set to be used as a model for allocation of new data sets.

Server This setting applies when creating files on the server's system (for example, with a PUT subcommand).

Client This setting applies when creating files on the client's system (for example, with a GET subcommand).

Syntax

```
►►──DCBDSN──name──────────────────────────────────────────────────────►◄
```

Parameters

name
The name of the data set to be used as a model for allocation of new data sets created with a STOR or MKDIR command.

Requirement: This data set name must be a fully qualified MVS data set name; z/OS UNIX file names are not allowed. There is no default.

Usage notes

- If specified or set to the default value, the following FTP.DATA statements, SITE command parameters, or locsite subcommand parameters override the DCB values from the model data set:
  - BLKSIZE
  - LRECL
  - RECFM
  - RETPD
- If you specify the MGMTCLASS statement, the retention period from the model data set can be overridden by the retention period specified by the SMS management class.
- When using a model DCB at the server, SENDSITE must be toggled off at the client. Otherwise, the SITE information sent automatically by the client overrides the value provided by the model DCB.
- BLKSIZE can also be specified with no value to allow the attributes from the model DCB to be used:

```
DCBDSN model.dcb
BLKSIZE
LRECL 0
RECFM
RETPD
```

Related topics

- “BLKSIZE (FTP client and server) statement” on page 756
- “DSNTYPE (FTP client and server) statement” on page 783
- “EATTR (FTP client and server) statement” on page 788
- “LRECL (FTP client and server) statement” on page 813
- “MGMTCLASS (FTP client and server) statement” on page 818
- “RECFM (FTP client and server) statement” on page 833
- “RETPD (FTP client and server) statement” on page 838
- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.

DCONNTIME (FTP client and server) statement

Use the DCONNTIME statement to define the amount of time that FTP waits attempting to close a data a data transfer before terminating the connection and reporting an error.

Server This setting specifies the time that the server waits on the client.

Client This setting specifies the time that the client waits on the server.

Syntax

```
   ┌─DCONNTIME 120──────┐
►►─┴─DCONNTIME seconds──┴──────────────────────────────────────────────►◄
```

Parameters

seconds
: The number of seconds FTP waits to receive notification that the data connection is closing. The valid range is 0 (DCONNTIME not used) or 15-86400. The default is 120.

Examples

Set the timer to 600 seconds:

```
DCONNTIME 600
```

Usage notes

If you specify 0 seconds, the DCONNTIME timer is disabled and FTP receives the FIN before closing the data connection.

DEBUG (FTP client and server) statement

Use the DEBUG statement to activate a specific trace type.

Restriction: Only one trace type can be activated for a DEBUG statement.

Server Traces are recorded on server's system for server processing.

Client Traces are recorded on client's system for client processing.

Syntax

```
►►──DEBUG──parameter─────────────────────────────────────────────────►◄
```

Parameters

FLO
: The FLO trace shows the flow of control within FTP. It is used to show which services of FTP are used for an FTP request.

CMD
: The CMD trace shows each command and the parsing of the parameters for the command.

PAR
: The PAR trace shows details of the FTP command parser. It is useful when debugging problems with the processing of command parameters.

INT
: The INT trace shows the details of the initialization and termination of the FTP session.

ACC
: The ACC trace shows the details of the login process.

UTL
: The UTL trace shows the processing of utility functions such as CD and SITE.

FSC(n)
: The FSC trace shows details of processing the file services server commands APPE, STOR, STOU, RETR, DELE, RNFR, and RNTO. For the client, it shows the details for subcommands, such as GET, PUT, APPEND, DELETE, and

RENAME. This trace allows you to specify levels of detail for the trace points. The level one tracing specified by entering FSC or FSC(1) is the level typically used unless more data is requested by the TCP/IP service group. *n* can be an integer between 1 and 8.

SEC
The SEC trace shows the processing of security functions such as TLS and GSSAPI negotiations.

SOC(n)
The SOC trace shows details of the processing during the setup of the interface between the FTP application and the network as well as details of the actual amounts of data that are processed. This trace allows you to specify levels of detail for the trace points. The level one tracing that is specified by entering SOC or SOC(1) is the level typically used unless more data is requested by the TCP/IP service group. *n* can be an integer between 1 and 8.

JES
The JES trace shows details of the processing for JES requests (that is, requests when SITE FILETYPE=JES is in effect).

Restriction: This parameter applies to the server only.

SQL
The SQL trace shows details of the processing for SQL requests (that is, requests when SITE or LOCSITE FILETYPE=SQL is in effect).

ALL
This value is used to set all of the trace points. Both the FSC and the SOC trace are set to level one when the ALL parameter is processed.

BAS
This value is used to set a select group of traces that offer the best overall details without the more excessive tracing some of the other traces provide. Specifying this value is the same as the following values:

- DEBUG CMD
- DEBUG INT
- DEBUG FSC
- DEBUG SOC

USERID (*filter_name*)
This parameter is used to filter the trace for user IDs matching the *filter_name* pattern. If the user ID matches the filter at the time the client logs in, tracing options are set to the current value of the options. Otherwise, no tracing options are set. The client can use the SITE command to set options after login if the initial ones are not appropriate. An example for the USERID filter is:

```
DEBUG USERID(USER33)
```

which activates the trace for a user if the user ID is USER33.

Restriction: This parameter applies to the server only.

IPADDR (*filter*)
This parameter is used to filter the trace for IP addresses matching the filter pattern. If the IP address matches the filter at the time the client connects, tracing options are set to the current value of the options. Otherwise, no tracing options are set. The client might use the SITE command to set options after connect if the initial ones are not appropriate. Examples of the IPADDR(*filter*) are:

```
DEBUG IPADDR(9.67.113.57)
DEBUG IPADDR(FEDC:BA98:7654:3210:FEDC:BA98:7654:3210)
```

The first example activates the trace for a client whose IP address is 9.67.113.57; the second activates the trace for a client whose IP address is FEDC:BA98:7654:3210:FEDC:BA98:7654:3210. If the filter is an IPv4 address, submasking can be indicated by using a slash followed by a dotted decimal submask. For example, 192.48.32.0/255.255.255.0 allows addresses from 192.48.32.00 to 192.48.32.255.

If the filter is an IPv6 address, network prefixing can be indicated by using a slash followed by a network prefix. For example, use FEDC:BA98::0/32 to indicate the prefix: FEDCBA98.

Restriction: This parameter applies to the server only.

DEBUGONSITE (FTP server) statement

Use the DEBUGONSITE statement to specify whether the FTP server accepts a SITE DEBUG command to change the general tracing options for the FTP session.

Syntax

```
    ┌─DEBUGONSITE FALSE───────┐
►►──┴─DEBUGONSITE──┬─TRUE──┬──┴──────────────────────────────────────────►◄
                   └─FALSE─┘
```

Parameters

TRUE
The server accepts a SITE DEBUG command from the client to change the general trace options for the current session.

FALSE
The server does not accept a SITE DEBUG command from the client. This is the default.

Related topics

- "DEBUG (FTP client and server) statement" on page 778
- See z/OS Communications Server: IP User's Guide and Commands for more information about the SITE subcommand.

DEST (FTP server) statement

Use the DEST statement to specify the NJE destination to which the files are routed when the server receives a STOR, STOU, or APPE command. Using the DEST statement allows you to send data sets to other users on machines connected on a network job entry (NJE) network rather than storing them at the server.

Syntax

```
►►──DEST──destination─────────────────────────────────────────────────►◄
```

Parameters

destination
: The NJE destination to which the files are routed when the server receives a STOR, STOU, or APPE command. The format for *destination* should be one of the following:

- userID@nodeID
- nodeID.userID
- nodeID
- DestID

There is no default.

Examples

Send files to user USER14 at system MVS1 instead of storing them in the server file system:

```
DEST USER14@MVS1
```

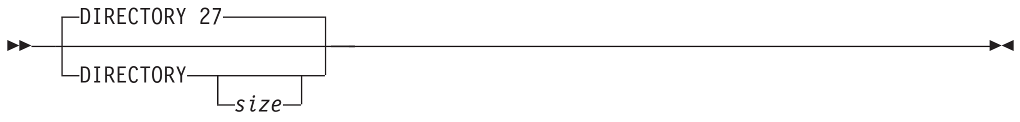
DIRECTORY (FTP client and server) statement

Use the DIRECTORY statement to specify the number of directory blocks to be allocated for the directory of a PDS.

Server This setting applies when creating files on the server's system (for example, with a PUT subcommand).

Client This setting applies when creating files on the client's system (for example, with a GET subcommand).

Syntax



Parameters

size
: The number of directory blocks to be allocated for the directory of a PDS. The valid range is 1 - 16 777 215 blocks (the operating system maximum). The default is 27.

Examples

Allocate a PDS with 15 directory blocks:

```
Directory 15
```

Specify DIRECTORY with no value to allow the directory information from an SMS dataclass to be used:

```
DIRECTORY
```

Usage notes

- If you specify no value for the *size*, FTP does not specify the number of directory blocks to be allocated for the directory of a PDS.
- You should specify no value for the *size* if the DATACLASS statement is specified and the directory from the SMS data class is to be used.

Related topics

- "DATACLASS (FTP client and server) statement" on page 770
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- "MGMTCLASS (FTP client and server) statement" on page 818
- "PDSTYPE (FTP client and server) statement" on page 826
- "STORCLASS (FTP client and server) statement" on page 877
- See the storage management subsystem (SMS) information in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.

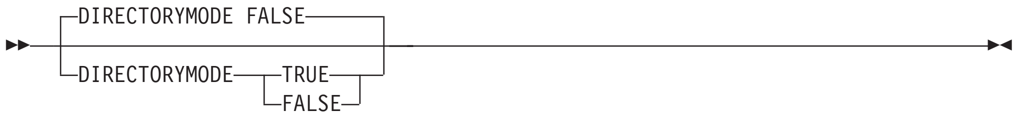
DIRECTORYMODE (FTP client and server) statement

Use the DIRECTORYMODE statement to specify whether only the data set qualifier immediately below the current directory is treated as an entry in the directory or if all data set qualifiers below the current directory are treated as entries in the directory.

Server This setting applies when issuing the MGET, LS, DIR, and MDELETE subcommands.

Client This setting applies when issuing the MPUT subcommand.

Syntax



Parameters

TRUE
Specifies that only the data set qualifier immediately below the current directory is treated as an entry in the directory.

FALSE
Specifies that all data set qualifiers below the current directory are treated as entries in the directory. This is the default.

Examples

If DIRECTORYMODE TRUE:

```
ftp> ls
200 Port request OK.
125 List started OK.
AREADME
BAILEY
BAILEY.MSYS.SPX001.I2.TEMP
BAILEY.TRANS
EZACIMJA
ISPF.ISPROF
XMLS
XX.AREADME
250 List completed successfully.
101 bytes received in 0.03 seconds (3.37 Kbytees/sec)
```

If DIRECTORYMODE FALSE:

```
ftp> ls
200 Port request OK.
125 List started OK.
AREADME
BAILEY
BAILEY
EZACIMJA
ISPF
XMLS
XX
250 List completed successfully.
51 bytes received in 0.03 seconds (1.07 Kbytees/sec)
```

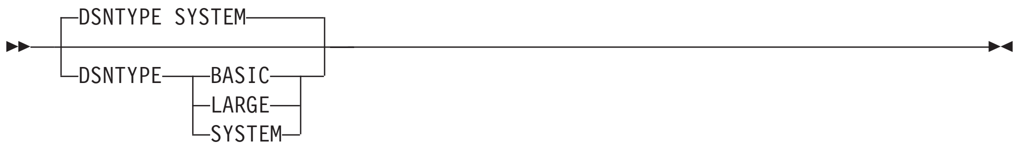
DSNTYPE (FTP client and server) statement

Use the DSNTYPE statement to specify whether FTP creates local physical sequential data sets as physical sequential basic format data sets or physical sequential large format data sets. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax



Parameters

BASIC
Allocates physical sequential data sets as physical sequential basic format data sets.

LARGE
Allocates physical sequential data sets as physical sequential large format data sets.

SYSTEM
Allocates physical sequential data sets with the SMS data class value, or the system default value.

Examples

To allocate new physical sequential data sets as physical large format data sets, code the following statement in FTP.DATA:

```
DSNTYPE  LARGE
```

Related topics:

- See the storage management subsystem (SMS) information in the z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "BLKSIZE (FTP client and server) statement" on page 756
- "DATACLASS (FTP client and server) statement" on page 770
- "EATTR (FTP client and server) statement" on page 788
- "LRECL (FTP client and server) statement" on page 813
- "PRIMARY (FTP client and server) statement" on page 830
- "RECFM (FTP client and server) statement" on page 833
- "RETPD (FTP client and server) statement" on page 838
- "SECONDARY (FTP client and server) statement" on page 845
- "SPACETYPE (FTP client and server) statement" on page 873
- "VOLUME (FTP client and server) statement" on page 895

DSWAITTIME (FTP client and server) statement

Use the DSWAITTIME statement to specify the number of minutes that FTP tries to access an MVS data set that could not be obtained because another job or process was holding the data set. FTP tries to access the data set approximately every minute for the number of minutes specified in the DSWAITTIME statement.

Restriction: The DSWAITTIME statement does not support tape data sets.

Server Specifies how many minutes the server tries to access to MVS data set.

Client Specifies how many minutes the client tries to access to MVS data set.

Rule: The FTP server reply that displays the holder and other useful information related to the MVS data set is issued only when REPLYSECURITYLEVEL is 0.

Syntax

```
     ┌─DSWAITTIME 0─────────┐
►►───┤                      ├──────────────────────────────────────────►◄
     └─DSWAITTIME ──minutes─┘
```

Parameters

minutes
The number of minutes to wait for an MVS data set to become available. Valid values are 0 (DSWAITTIME not used) or 1 - 14400. The default is 0.

Rule: If DSWAITTIME is set to 0, the timer is not set, and only one attempt is made to access an MVS data set.

Guidelines:

- The FTP server ignores the DSWAITTIME configuration option for RENAME FROM (RNFR), RENAME TO (RNTO), and DELETE (DELE) commands.
- If the DSWAITTIME configuration option is not 0, also specify the DATAKEEPALIVE configuration option.
- If you experience control connection timeouts while the server is waiting for access to an MVS data set, try configuring a nonzero value for

FTPKEEPALIVE at the client or the server. If keepalive packets do not prevent FTP control connection timeouts, configure a smaller DSWAITTIMEREPLY value.

Examples

Use the following code to set the data set wait time to 10 minutes:

```
DSWAITTIME 10
```

Related topics

- “DSWAITTIMEREPLY (FTP server) statement”
- “DATAKEEPALIVE (FTP client and server) statement” on page 773
- “FTPKEEPALIVE (FTP client and server) statement” on page 797
- “REPLYSECURITYLEVEL (FTP server) statement” on page 836

DSWAITTIMEREPLY (FTP server) statement

Use the DSWAITTIMEREPLY statement to specify how often to send the following reply message to the client while the FTP server is waiting for access to an MVS data set.

```
125- Data set access will be retried in 1 minute intervals - number attempts remaining
```

Results:

- The server always issues the following reply at one-minute intervals while waiting for access to a data set for the amount of time specified by the DSWAITTIME configuration option.

  ```
  125- FTP Server unable to obtain usage use of data set
  which is held by asid jobname accessmode on qname
  ```

  By default, the server also sends the following reply to the client at one-minute intervals.

  ```
  125- Data set access will be retried at 1 minute intervals - number attempts remaining
  ```

  Set the DSWAITTIMEREPLY value to a number smaller than 60 to cause the server to send this reply more frequently.
- The DSWAITTIMERREPLY value is applied only when you have configured a DSWAITTIME value that is greater than zero.

Tip: Coding a DSWAITTIMEREPLY value that is smaller than 60 can prevent an FTP client connection from timing out while the server is waiting for a data set.

Syntax

```
     ┌─DSWAITTIMEREPLY 60───────┐
►►───┤                          ├──────────────────────────────────────►◄
     └─DSWAITTIMEREPLY──seconds─┘
```

Parameters

seconds
: The number of seconds between reply messages that the server sends to the client while it is waiting for access to an MVS data set. The valid range is 15 - 60. The default value is 60. The reply message that is sent:

  ```
  125- Data set access will be retried at 1 minute intervals - number attempts remaining
  ```

Examples

In this example, the DSWAITTIMEREPLY value is 30. The following reply message is issued every 30 seconds:

```
125- Data set access will be retried at 1 minute intervals - number attempts remaining
```

If the DSWAITTIMEREPLY value is 3, the following would be the actual output:

```
125- FTP Server unable to obtain EXCLUSIVE use of
USER.TEST.DATA which is held by: 0035 USER2
EXCL on SYSDSN
125- Data set access will be retried in 1 minute
intervals - 3 attempts remaining
125- Data set access will be retried in 1 minute
intervals - 3 attempts remaining
125- FTP Server unable to obtain EXCLUSIVE use of
USER.TEST.DATA which is held by: 0035 USER2
EXCL on SYSDSN
125- Data set access will be retried in 1 minute
intervals - 2 attempts remaining
125- Data set access will be retried in 1 minute
intervals - 2 attempts remaining
125- FTP Server unable to obtain EXCLUSIVE use of
USER.TEST.DATA which is held by: 0035 USER2
EXCL on SYSDSN
125- Data set access will be retried in 1 minute
intervals - 1 attempts remaining
125- Data set access will be retried in 1 minute
intervals - 1 attempts remaining
125- FTP Server unable to obtain EXCLUSIVE use of
USER.TEST.DATA which is held by: 0035 USER2
EXCL on SYSDSN
125 Data set USER2.TEST.DATA is not available
```

Related topics

- "DSWAITTIME (FTP client and server) statement" on page 784
- "FTPKEEPALIVE (FTP client and server) statement" on page 797

DUMP (FTP client and server) statement

Use the DUMP statement to activate an extended trace.

Restriction: Only one dump parameter can be specified for a DUMP statement.

Server Extended traces are recorded on server's system for server debugging.

Client Extended traces are recorded on client's system for client debugging.

Syntax

```
►►──DUMP──parameter──────────────────────────────────────────────►◄
```

Parameters

n Specifies the ID number of a specific extended trace point that is to be activated in the FTP code. The number is an integer in the range 1 - 99.

FSC
Activates all of the extended trace points in the file services code.

JES
Activates all of the extended trace points in the JES services code.

Restriction: This applies to the server only.

SOC
Activates all of the extended trace points in the network services code.

SQL
Activates all of the extended trace points in the SQL services code.

ALL
This parameter is used to set all of the trace points. It sets dump IDs 1 to 99.

USERID (*filter_name***)**
This parameter is used to filter the extended trace for user IDs matching the *filter_name* pattern. If the user ID matches the filter at the time the client logs in, tracing options are set to the current value of the options. Otherwise, no extended tracing options are set. The client might use the SITE command to set options after login if the initial ones are not appropriate. An example for the USERID filter is:

```
DUMP USERID(USER33)
```

which activates the dumpID trace for a user if the user ID is USER33.

Restriction: This applies to the server only.

IPADDR (*filter/subnet mask***)**
This parameter is used to filter the extended trace for IP addresses matching the filter pattern. If the IP address matches the filter at the time the client connects, extended tracing options are set to the current value of the options. Otherwise, no extended tracing options are set. The client might use the SITE command to set options after connect if the initial ones are not appropriate. Examples of the IPADDR filter are:

```
DUMP IPADDR(9.67.113.57)
DUMP IPADDR(FEDC:BA98:7654:3210:FEDC:BA98:7654:3210)
```

The first example activates the extended traces for a client whose IP address is 9.67.113.57; the second activates the extended traces for a client whose IP address is FEDC:BA98:7654:3210:FEDC:BA98:7654:3210.

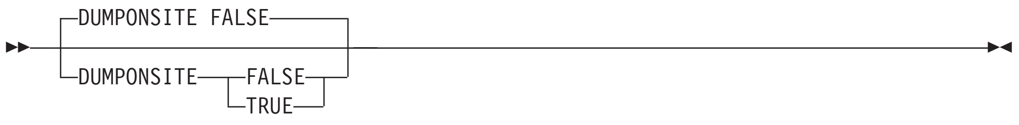
If the filter is an IPv4 address, submasking can be indicated by using a slash followed by a dotted decimal submask. For example, 192.48.32/255.255.255.0 allows addresses from 192.48.32.00 to 192.48.32.255. If the filter is an IPv6 address, network prefixing can be indicated by using a slash followed by a network prefix. For example, use FEDC:BA98::0/32 to indicate the prefix: FEDCBA98.

Restriction: This applies to the server only.

DUMPONSITE (FTP server) statement

Use the DUMPONSITE statement to specify whether the FTP server accepts a SITE DUMP command to change the extended tracing options for the FTP session.

Syntax



Parameters

TRUE
: The FTP server allows an FTP client to change the extended trace options with a SITE DUMP command.

FALSE
: The FTP server does not allow an FTP client to change the extended trace options with a SITE DUMP command. This is the default.

Related topics

- "DUMP (FTP client and server) statement" on page 786
- See z/OS Communications Server: IP User's Guide and Commands for more information about the SITE subcommand.

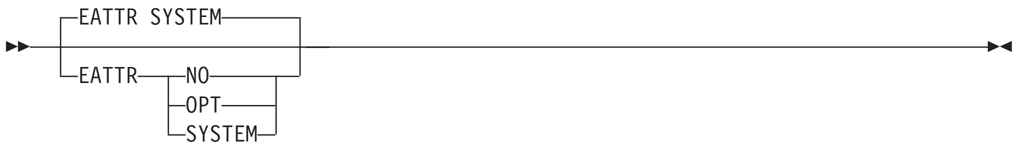
EATTR (FTP client and server) statement

Use the EATTR statement to specify whether new data sets can have extended attributes and whether the data sets can reside in the EAS of an EAV. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax



Parameters

NO The new data set cannot reside in the EAS, and its VTOC entry cannot have extended attributes.

OPT
: The new data set can reside in the EAS, and its VTOC entry can have extended attributes if the volume supports them.

SYSTEM
: The new data set will use the SMS data class EATTR value. If no SMS data class is defined, or if the data class contains no EATTR specification, the data set will be allocated with the system default. This is the default.

Examples

To allow new data sets to have extended attributes if the volume supports them, and to allow new data sets to reside in the EAS, code the following statement in FTP.DATA:

```
EATTR  OPT
```

Related topics:

- See the storage management subsystem (SMS) information in the z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "BLKSIZE (FTP client and server) statement" on page 756
- "DATACLASS (FTP client and server) statement" on page 770
- "DCBDSN (FTP client and server) statement" on page 776
- "DIRECTORY (FTP client and server) statement" on page 781
- "DSNTYPE (FTP client and server) statement" on page 783
- "LRECL (FTP client and server) statement" on page 813
- "PDSTYPE (FTP client and server) statement" on page 826
- "PRIMARY (FTP client and server) statement" on page 830
- "RECFM (FTP client and server) statement" on page 833
- "RETPD (FTP client and server) statement" on page 838
- "SECONDARY (FTP client and server) statement" on page 845
- "SPACETYPE (FTP client and server) statement" on page 873
- "VOLUME (FTP client and server) statement" on page 895

EMAILADDRCHECK (FTP server) statement

Use the EMAILADDRCHECK statement to control the extent to which the FTP server validates e-mail addresses entered by FTP clients while logging in to the FTP server.

Restriction: This statement is meaningful only when ANONYMOUSLEVEL is 3 or greater.

Syntax

```
    ┌─EMAILADDRCHECK NO──────────┐
►►──┤                            ├──────────────────────────────────────────►◄
    └─EMAILADDRCHECK───┬─NO──────┬┘
                       ├─FAIL────┤
                       └─WARNING─┘
```

Parameters

NO The FTP server does not validate the e-mail address entered by the FTP client. Whatever the user entered is accepted and the user can log in. This is the default.

FAIL
The FTP server verifies that the e-mail address entered by the FTP client is a valid e-mail address before allowing the user to log in. The FTP server rejects the login if the e-mail address is not valid.

WARNING
The FTP server inspects the e-mail address entered by the FTP client. Any value the client enters is accepted as valid; however, the FTP server returns a warning reply to the client if the e-mail address is not plausible. In either case, the FTP server allows the FTP client to log in.

Examples

To ensure that only anonymous users entering valid e-mail addresses are allowed successful login, set the following parameter in FTP.DATA:

```
EMAILADDRCHECK FAIL; Requires anonymous users to enter a valid email address.
```

Usage notes

The FTP server prompts anonymous users for an e-mail address instead of a password when ANONYMOUSLEVEL is 3.

Related topics

- "ANONYMOUS (FTP server) statement" on page 738
- "ANONYMOUSLEVEL (FTP server) statement" on page 748
- "The FTCHKPWD user exit" on page 707

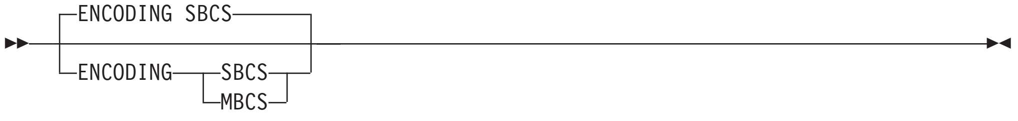
ENCODING (FTP client and server) statement

Use the ENCODING statement in the server and client FTP.DATA to indicate the type of data encoding on the network. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server Specifies to the server whether to use single or double-byte code pages.

Client Specifies to the client whether to use single or double-byte code pages.

Syntax



Parameters

SBCS
Specifies single byte encoding. Code pages are specified by way of the SBDATACONN statement. This is the default value.

MBCS
Specifies multibyte encoding. Code pages are specified by way of the MBDATACONN statement.

Rule: The data transfer Type must be ASCII to enable multibyte translation when ENCODING=MBCS is set.

Tip: The type is always ASCII when the client initially logs into the server.

Server
- The data transfer Type remains ASCII until the server receives a **Type** command from the client
- You can send a STAT command to the server to verify the Type setting by issuing the **stat** subcommand from the z/OS FTP client, or by issuing a QUOTE STAT command from any FTP client.

Client

- Certain subcommands, such as TYPE, BIG5, and others, change the data transfer Type.
- You can use the LOCSTAT subcommand to verify the Type setting.
- Use the TYPE subcommand to restore Type to ASCII

Examples

To indicate that data encoding was specified using MBDATACONN statement, use the following code:

```
ENCODING MBCS
```

Related topics

- "MBDATACONN (FTP client and server) statement" on page 815
- "MBSENDEOL statement (FTP client and server) statement" on page 817
- "SBDATACONN (FTP client and server) statement" on page 840
- "SBSENDEOL statement (FTP client and server) statement" on page 841

EPSV4 (FTP client) statement

Use the EPSV4 statement to direct the FTP client to use EPSV and EPRT commands on IPv4 sessions. The locsite subcommand is also available to set this parameter.

Syntax

```
    ┌─EPSV4 FALSE──────┐
►►──┤                  ├──────────────────────────────────────────────►◄
    └─EPSV4──┬─FALSE─┬─┘
             └─TRUE──┘
```

Parameters

FALSE
Prevents the client from using EPRT and EPSV commands on IPv4 sessions. This is the default.

TRUE
Directs the client to use EPRT and EPSV commands on IPv4 sessions.

Usage notes

EPRT and EPSV commands are described in RFC 2428. If the server rejects an EPRT or EPSV command during the session, the client stops sending EPRT and EPSV to that server regardless of how you have set EPSV4.

Guideline: If your client has trouble establishing a data connection on an IPv4 security protected, encrypted session through an NAT firewall, coding EPSV4 TRUE in the client's FTP.DATA can help.

Restrictions:

- The FTP server ignores this statement.

- Socksified sessions use PASV or PORT commands to establish data connections, as specified by the FWFRIENDLY setting. When EPSV4 is TRUE, the client attempts EPSV but never EPRT to establish a socksified data connection.
- Some FTP servers support EPRT and EPSV commands, but do not reply as described in RFC 2428. If the FTP server reply to EPSV or EPRT does not conform to RFC 2428, the client reacts as if the server has rejected the command.
- RFC 2428 stipulates EPSV is the preferred command to establish data connections. Therefore, when EPSV4 is TRUE, the client tries EPSV regardless of how you have set FWFRIENDLY. The client uses EPRT only to set up a data connection for proxy transfer.

Examples

To direct the client to use EPSV and EPRT commands on IPv4 FTP sessions, use the following code:

```
EPSV4 TRUE
```

Related topics

- “FWFRIENDLY (FTP client) statement” on page 799
- “PASSIVEIGNOREADDR (FTP client) statement” on page 824

EXTENSIONS (FTP client and server) statement

Use the EXTENSIONS statement to enable FTP to support FTP extensions not described in RFC 959.

Syntax

```
►►──┬──────────────────────────┬─────────────────────────────────────►◄
    ├─EXTENSIONS AUTH_GSSAPI───┤
    ├─EXTENSIONS SIZE──────────┤
    ├─EXTENSIONS MDTM──────────┤
    ├─EXTENSIONS UTF8──────────┤
    ├─EXTENSIONS REST_STREAM───┤
    └─EXTENSIONS AUTH_TLS──────┘
```

Parameters

AUTH_GSSAPI
Specifies that GSSAPI authentication is supported. The server supports receiving the AUTH command with GSSAPI. AUTH_GSSAPI is supported for IPv4 connections only.

Restriction: This parameter applies to the server only.

SIZE
Enables the FTP Server to respond to the SIZE command. SIZE is supported for z/OS UNIX files only when the data transfer type is image, ASCII, or EBCDIC, the structure is file, and the data transfer mode is stream. If an FTP client requests MDTM or SIZE information for an MVS data set, or for any other unsupported file, the server returns an FTP reply code and error message instead of the requested information.

Restriction: This parameter applies to the server only.

MDTM

Enables the FTP Server to respond to the MDTM command. MDTM is supported for z/OS UNIX files only.

Restriction: This parameter applies to the server only.

UTF8

Enables the FTP server to respond to the LANG command, and to use UTF–8 encoding of pathnames on the control connection. The server ignores configuration options that direct it to use a specific code page on the control connection, as well as SITE commands that specify a specific code page on the control connection. The server initializes the control connection to use 7-bit ASCII until a LANG command from the client directs it to use UTF-8 encoding of pathnames. The only language supported by the server is United States English.

When the client has EXTENSIONS UTF8 encoded in FTP.DATA, the client has the language and subcommands available. Configuration options that direct the client to use a specific code page on the control connection, as well as LOCSITE commands that specify a specific code page on the control connection, are ignored. Initially the client uses 7-bit ASCII on the control connection. During client login, the client queries the server to determine whether it supports UTF-8 encoding. If so, it uses UTF-8 encoding of pathnames on the control connection.

Restriction: This parameter applies to both the client and the server.

REST_STREAM

Enables the FTP server to restart stream mode file transfers. The server ignores EXTENSIONS REST_STREAM unless EXTENSIONS SIZE is also coded, because stream restarts rely on the SIZE command.

Restriction: This parameter applies to the server only.

AUTH_TLS

Specifies that TLS authentication is supported. The server supports receiving the AUTH command with the following values:

- TLS: When the server successfully processes the AUTH TLS command and completes the handshake with the FTP client, the control connection is protected by TLS.
- TLS-C: When the server successfully processes the AUTH TLS-C command and completes the handshake with the FTP client, the control connection is protected by TLS.
- TLS-P: When the server successfully processes the AUTH TLS-P command and completes the handshake with the FTP client, the control connection is protected by TLS. The server also implicitly protects all data connections.
- SSL: When the server successfully processes the AUTH SSL command and completes the handshake with the FTP client, the control connection is protected by TLS.The server also implicitly protects all data connections.

Restriction: This parameter applies to the server only.

Results:

- This parameter also enables server support for the PROT and PBSZ commands.
- Server support for TLS-secured sessions is affected by the TLSRFCLEVEL setting.

Examples

```
EXTENSIONS SIZE
```

```
EXTENSIONS MDTM
```

Usage notes

- The EXTENSIONS statement has no default value.
- If you do not include an EXTENSIONS statement in FTP.DATA, no extensions to RFC 959 are recognized.
- Unlike other FTP.DATA statements, EXTENSIONS statements are cumulative. If you include an EXTENSIONS SIZE statement in FTP.DATA and also an EXTENSIONS MDTM statement, the FTP server receives both the SIZE and MDTM commands.
- The only way to disable an EXTENSIONS statement is to remove that statement from FTP.DATA. You can remove a statement by changing it to a comment or by deleting the statement.
- The SIZE and MDTM commands are not part of RFC 959. They are proposed commands described by an Internet-Draft published by the IETF (Internet Engineering Task Force). Because these commands are not part of an RFC, the FTP server supports them only if FTP.DATA includes EXTENSIONS statements to explicitly enable them.

Related topics

- "TLSRFCLEVEL (FTP client and server) statement" on page 880
- See z/OS Communications Server: IP Configuration Guide for more information about customizing TLS and Kerberos.

FIFOIOTIME (FTP client and server) statement

Use the FIFOIOTIME statement to set a timeout for reading and writing to a UNIX named pipe. This timeout is the maximum length of time FTP waits for I/O to a UNIX named pipe to complete. You can use the SIte and LOCSIte subcommands to set this value.

Server Specifies how long the server waits for reads from and writes to a UNIX named pipe to complete.

When you are retrieving data from a named pipe in the FTP server file system, this statement specifies the length of time the server waits for reads from the named pipe to complete.

When you are storing data into a named pipe in the FTP server file system, this statement specifies the length of time the server waits for writes to the named pipe to complete.

If no data is written to or read from the named pipe in the FIFOIOTIME interval, the FTP server fails the file transfer.

Tip: Setting FIFOIOTIME to a small value interrupts the server needlessly. This can have a deleterious impact on FTP performance.

Client Specifies the length of time that the client waits for reads from and writes to a UNIX named pipe to complete.

When you are sending a file from a named pipe in the FTP client file system to the FTP server, this statement specifies the length of time that the client waits for reads from the named pipe to complete.

When you are getting a file from the FTP server and storing it into a named pipe in the FTP client server file system, this statement specifies the length of time the client waits for writes to the named pipe to complete.

If no data is written to or read from the named pipe in the FIFOIOTIME interval, the FTP client fails the file transfer.

Syntax

```
   ┌─FIFOIOTIME 20──────┐
►►─┤                    ├────────────────────────────────────────────────►◄
   └─FIFOIOTIME seconds─┘
```

Parameters

seconds
The number of seconds in the range 1 - 86 400. The default is 20.

Examples

Use the following code to set the timer to 60 seconds:

```
FIFOIOTIME 60
```

Related topics

- “FIFOOPENTIME (FTP client and server) statement”
- “UNIXFILETYPE (FTP client and server) statement” on page 891

FIFOOPENTIME (FTP client and server) statement

Use the FIFOOPENTIME statement to define the length of time that FTP waits after attempting to open UNIX named pipe before reporting an error. You can use the SIte and LOCSIte subcommands to set this value.

Server This setting specifies the length of time that the server waits for an open of a UNIX named pipe to complete.

Client This setting specifies the length of time that the client waits for an open of a UNIX named pipe to complete.

Syntax

```
   ┌─FIFOOPENTIME 60──────┐
►►─┤                      ├──────────────────────────────────────────────►◄
   └─FIFOOPENTIME seconds─┘
```

Parameters

seconds
The number of seconds that FTP waits for an open of a UNIX named pipe to complete. Valid values are in the range 1 - 86 400. The default is 60.

Examples

Use the following code to set the timer to 600 seconds:

```
FIFOOPENTIME 600
```

Related topics

- "FIFOIOTIME (FTP client and server) statement" on page 794
- "UNIXFILETYPE (FTP client and server) statement" on page 891

FILETYPE (FTP client and server) statement

Use the FILETYPE statement to specify the method of operation for FTP.

Syntax

```
   ┌─FILETYPE SEQ──────────┐
►►─┴─FILETYPE──┬─JES─┬─────┴──────────────────────────────────────────►◄
               ├─SEQ─┤
               └─SQL─┘
```

Parameters

JES
Remote job submission.

Restriction: This parameter applies to the server only.

SEQ
MVS data sets or z/OS UNIX files. SEQ is the method of operation supported by all FTP platforms. This is the default.

SQL
SQL query function. SQL method affects the RETR command at the server and the PUT subcommand at the client.

Examples

Set the operational method to SQL:

```
Filetype SQL
```

Usage notes

- SQL pertains to z/OS platform only. For more information about the effects on command processing when FILETYPE is SQL, see z/OS Communications Server: IP User's Guide and Commands.
- When the SQL method is in specified for the server, it affects the RETR command only. When the SQL method is specified for the client, it affects the STOR command only.
- JES pertains to the z/OS platform only and is valid only in the FTP.DATA file for a server. For more information about the effects on command processing at the server when the server's FILETYPE is JES, see z/OS Communications Server: IP User's Guide and Commands.
- JES method affects the STOR, LIST, RETR, and NLST commands.

Related topics

- "ANONYMOUS (FTP server) statement" on page 738
- "ANONYMOUSLEVEL (FTP server) statement" on page 748
- "ANONYMOUSFILETYPEJES (FTP server) statement" on page 742
- "ANONYMOUSFILETYPESEQ (FTP server) statement" on page 743

- “ANONYMOUSFILETYPESQL (FTP server) statement” on page 743
- “DB2 (FTP client and server) statement” on page 774
- “DB2PLAN (FTP cilent and server) statement” on page 775
- “JESENTRYLIMIT (FTP server) statement” on page 802
- “JESLRECL (FTP server) statement” on page 806
- “JESPUTGETTO (FTP server) statement” on page 807
- “JESGETBYDSN (FTP server) statement” on page 803
- “JESINTERFACELEVEL (FTP server) statement” on page 804
- z/OS Communications Server: IP Configuration Guide for information about JESINTERFACELEVEL

FTPKEEPALIVE (FTP client and server) statement

Use the FTPKEEPALIVE statement to define the control connection keepalive timer value in seconds. This sets a socket level keepalive timer for the control connection. This allows the keepalive mechanism to send a packet on the idle control connection every FTPKEEPALIVE seconds, and avoid the firewall timing out the control connection.

Server Specifies how often the server sends a keepalive packet.

Client Specifies how often the client sends a keepalive packet.

Syntax

```
    ┌─FTPKEEPALIVE 0───────┐
►►──┼──────────────────────┼──────────────────────────────────────────────►◄
    └─FTPKEEPALIVE seconds─┘
```

Parameters

seconds
: The number of seconds before a keepalive packet is sent out on the FTP control connection. The valid range is 0 (FTPKEEPALIVE not used) or 60 - 86 400. The default is 0.

Examples

Set the FTP keepalive timer to 60 seconds:

```
FTPKEEPALIVE 60
```

Usage notes

If you specify 0 seconds, the FTPKEEPALIVE timer is disabled and the only keepalive packets that flow on the control connection would be controlled by whatever interval for keepalive packets you have configured in the stack.

FTPLOGGING (FTP server) statement

Use the FTPLOGGING statement to indicate whether the FTP server should log FTP server activity. The following types of activities are logged:

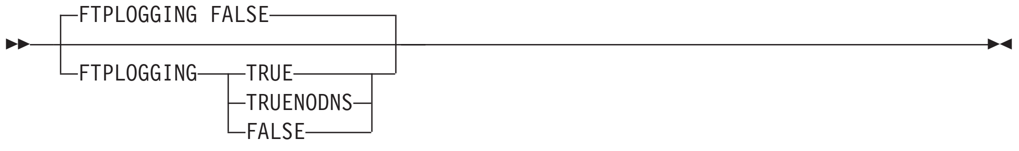
- Connectivity
- Authentication

- Access
- Allocation
- Deallocation
- Data transfer
- JES job submission
- SQL query
- Abnormal end
- Confidence of success level assigned to each file transfer when CHKCONFIDENCE is coded

The activities are logged in the SYSLOGD file. Each logging entry has a message number.

FTPLOGGING controls logging for non-anonymous user.

Syntax



Parameters

TRUE
The FTP server should log FTP session activity.

Tip: If TRUE is used, a long delay in login processing might occur because the FTP server issues a DNS query to resolve the remote host IP address.

TRUENODNS
The FTP server should log FTP session activity, however the client hostname lookup done during connection initiation is disabled. Message EZYFS50I contains UNKNOWN for the host name.

FALSE
The FTP server should not log FTP session activity.

Examples

To request that the FTP server log session activity:

```
FTPLOGGING TRUE
```

Usage notes

- Each activity logging message has a message number within the range of EZYFS50 to EZYFS95.
- If FTPLOGGING is TRUE, connectivity, authentication, and access activity log entries are made for all sessions because the server does not know whether the login is anonymous or not.

Related topics

- "CHKCONFIDENCE statement (FTP client and server) statement" on page 760

- See “ANONYMOUSFTPLOGGING (FTP server) statement” on page 744 to control logging for an anonymous user.

FWFRIENDLY (FTP client) statement

Use the FWFRIENDLY statement to specify how data connections are to be set up between the client and the server.

Syntax

```
   ┌─FWFRIENDLY FALSE──────┐
►►─┼───────────────────────┼──────────────────────────────────────────────►◄
   └─FWFRIENDLY──┬─TRUE──┬─┘
                 └─FALSE─┘
```

Parameters

TRUE
Specifies that the FTP client is firewall-friendly. This means that data connections are set up from the FTP client to the FTP server.

FALSE
Specifies that the FTP client is not firewall-friendly. This means that data connections are set up from the FTP server to the FTP client. This is the default.

Examples

```
FWFRIENDLY  TRUE      ; FTP client is firewall-friendly
```

Usage notes

When the connection to the server is IPv6, data connections are set up from client to the server regardless of the FWFRIENDLY setting.

Related topics

- “EPSV4 (FTP client) statement” on page 791
- “PASSIVEIGNOREADDR (FTP client) statement” on page 824

HFSINFO (FTP server) statement

Use the HFSINFO statement to specify a file containing welcome messages specific to each FTP server directory visited by an FTP user. In contrast to FTP users that are logged in as anonymous users, this statement affects only known users.

Syntax

```
►►──HFSInfo──file-mask──────────────────────────────────────────────────►◄
```

Parameters

file-mask
The file-mask is a z/OS UNIX file mask used to find a z/OS UNIX information file for known users. The file mask can contain wildcards or it can be a complete file name. When a user changes directories, a search is

conducted with the specified mask. The contents of the first file found is returned to the FTP client and is displayed to the end user. If no file is found matching the specified mask, no information is displayed to the end user.

Restriction: Wildcards work only when an asterisk (*) is placed after a string of characters.

Examples

Use the following code to direct the FTP server to search each directory to which a named FTP client changes, for a file matching the pattern msg*. Each time a named FTP client changes directory, the FTP server searches the target directory for files matching the file-mask msg*. The contents of the first matching file in each directory is returned to the FTP client.

```
HFSINFO msg* ; Real user HFS info file-mask
; login
```

Usage notes

The default value of HFSINFO is <null>, meaning no welcome messages are displayed.

Related topics

- "ADMINEMAILADDRESS (FTP server) statement" on page 738
- "ANONYMOUSHFSINFO (FTP server) statement" on page 747
- "ANONYMOUSLOGINMSG (FTP server) statement" on page 750
- "ANONYMOUSMVSINFO (FTP server) statement" on page 751
- "BANNER (FTP server) statement" on page 756
- "LOGINMSG (FTP server) statement" on page 813
- "MVSINFO (FTP server) statement" on page 820

INACTIVE (FTP Server) statement

Use the INACTIVE statement to set the inactivity timer to a specified number of seconds. Any control connection that is inactive for the amount of time specified on this statement is closed by the server.

Syntax

```
   ┌─INACTIVE 300──────┐
►►─┤                   ├──────────────────────────────────────────────►◄
   └─INACTIVE──seconds─┘
```

Parameters

seconds
: The number of seconds to which the inactivity timer is set. The valid range is 0 - 86 400. The default is 300. A value of 0 indicates no inactivity time is enabled, and the connection does not time out.

Examples

Set the inactivity timer to 30 seconds:

```
INACTIVE 30
```

Usage notes

- This value has no effect on the data connections. To specify a timeout value for the data connection, use the INTERVAL parameter of the TCPCONFIG statement in PROFILE.TCPIP. See the FTP configuration process in z/OS Communications Server: IP Configuration Guide for details.
- Specifying an INACTIVE value of zero can result in idle sessions remaining open indefinitely, which consumes system resources in an unproductive way. Code a nonzero value for INACTIVE to ensure that idle, unproductive sessions eventually expire.

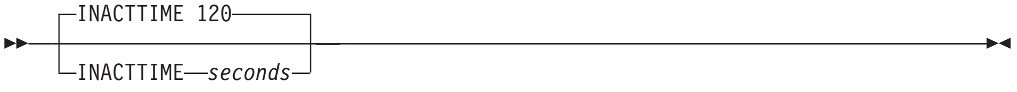
Related topics

- "FTPKEEPALIVE (FTP client and server) statement" on page 797

INACTTIME (FTP client) statement

Use the INACTTIME statement to specify the amount of time the FTP client waits for an expected response from the server, on either the control or the data connection, before closing the session. Data transfer times that exceed this value does not cause session termination unless the time between data packet arrivals exceeds this value.

Syntax



Parameters

seconds
The number of seconds to which the timer is set. The valid range is 0 (INACTTIME not used) or 15-86400. The default is 120 seconds.

Examples

```
INACTTIME  160  ; wait 160 seconds
```

Usage notes

None

Related topics

See the FTP command and FTP environment information in z/OS Communications Server: IP User's Guide and Commands.

ISPFSTATS (FTP client and server) statement

Use the ISPFSTATS statement to allow FTP to create and maintain statistics for partitioned data set members. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server This setting applies when creating or updating data sets on the server's system.

Client This setting applies when creating or updating data sets on the client's system.

Syntax

```
    ┌─ISPFSTATS FALSE────────┐
►►──┼────────────────────────┼────────────────────────────────────────►◄
    └─ISPFSTATS───┬─TRUE──┬──┘
                  └─FALSE─┘
```

Parameters

FALSE
FTP does not create statistics if the file does not already exist or does exist but does not have statistics. If the file already exists and contains statistics, FTP updates the statistics and sends the reply indicating the behavior.

TRUE
FTP creates or updates the statistics.

Examples

FTP creates the statistics:

```
ISPFSTATS TRUE
```

Usage notes

- The ISPFSTATS statement is ignored for sequential data sets; it applies to PDS and PDSE data sets. The record format must be either variable or fixed, and the record length must be less than 256.
- Transferring PDS member to PDS member in block mode or in compress mode differs in behavior from transferring in stream mode. If the user wants to preserve the statistics of the PDS member that already has statistics, and have the same statistics copied over to targeted PDS member, transferring in block mode or in compress mode is preferred.

JESENTRYLIMIT (FTP server) statement

Use the JESENTRYLIMIT statement to specify the number of entries that can be displayed concurrently through a LIST or NLST command when FILETYPE=JES and JESINTERFACELEVEL=2. You can also use the SITE command to set this keyword.

Syntax

```
     ┌─JESENTRYLIMIT 200───┐
►►───┴─JESENTRYLIMIT value─┴──────────────────────────────────────────────►◄
```

Parameters

value
    A numeral in the range of 1 - 1 024.

Examples

The following example illustrates a JESENTRYLIMIT of 10:

```
dir
 EZA1701I >>> PORT 127,0,0,1,4,10
 200 Port request OK.
 EZA1701I >>> LIST
 125 List started OK for JESJOBNAME=USER1*, JESSTATUS=ALL and JESOWNER=USER1
 EZA2284I JOBNAME  JOBID    OWNER    STATUS CLASS
 EZA2284I USER1    TSU00025 USER1    OUTPUT TSU      ABEND=222 3 spool files
 EZA2284I USER1A   JOB00209 USER1    OUTPUT A        ABEND=806 3 spool files
 EZA2284I USER1    JOB00201 USER1    OUTPUT A        RC=0000 5 spool files
 EZA2284I USER1J   JOB00208 USER1    OUTPUT A        (JCL error) 3 spool files
 EZA2284I USER1    JOB00193 USER1    OUTPUT A        RC=0000 5 spool files
 EZA2284I USER1A   JOB00200 USER1    OUTPUT A        ABEND=806 3 spool files
 EZA2284I USER1    JOB00179 USER1    OUTPUT A        RC=0000 5 spool files
 EZA2284I USER1    JOB00166 USER1    OUTPUT A        RC=0000 5 spool files
 EZA2284I USER1J   JOB00199 USER1    OUTPUT A        (JCL error) 3 spool files
 EZA2284I USER1A   JOB00187 USER1    OUTPUT A        ABEND=806 3 spool files
 250-JESENTRYLIMIT of 10 reached.  Additional entries not displayed
 250 List completed successfully.
 EZA1460I Command:
```

Usage notes

- If JESENTRYLIMIT is not specified in FTP.DATA, the default is 200.
- JESENTRYLIMIT is valid only when JESINTERFACELEVEL is set to 2.

Related topics

- “FILETYPE (FTP client and server) statement” on page 796
- “JESGETBYDSN (FTP server) statement”
- “JESINTERFACELEVEL (FTP server) statement” on page 804

JESGETBYDSN (FTP server) statement

Use the JESGETBYDSN statement to specify how to use the foreign file name when retrieving a file with a value of FILETYPE=JES.

When the JESGETBYDSN statement value FALSE is coded or set to the default value, the foreign file specified when retrieving a file with FILETYPE=JES is read from the MVS system, submitted to JES as a batch job, and its output is retrieved to the client.

When the JESGETBYDSN statement value TRUE is coded, the foreign file specified when retrieving a file with FILETYPE=JES is read as a JES spool file data set name, and its output retrieved to the client. The JES spool file data set name is the same format as an MVS data set name, but it is a case-sensitive JES data set name. The JES data set name for a job can be found using SDSF on the Job Data Set panel (JDS). See z/OS SDSF Operation and Customization for more information about JES data set names.

Syntax

```
   ┌─JESGETBYDSN FALSE──────┐
►►─┤                        ├────────────────────────────────────────────────►◄
   └─JESGETBYDSN──┬─TRUE──┬─┘
                  └─FALSE─┘
```

Parameters

FALSE
Specifies that the foreign file specified when retrieving a file with FILETYPE=JES is read from the MVS system, submitted to JES as a batch job, and its output is retrieved to the client. This is the default setting.

TRUE
Specifies that the foreign file specified when retrieving a file with FILETYPE=JES is read as a JES spool file data set name and its output is retrieved to the client.

Examples

The following example illustrates a JESGETBYDSN of FALSE:

```
JESGETBYDSN FALSE
```

Rule: The JESGETBYDSN statement only has meaning when FILETYPE=JES is specified and when JESINTERFACELEVEL 2 is coded in the FTP server's FTP.DATA file.

Related topics

- "ANONYMOUSFILETYPEJES (FTP server) statement" on page 742
- "FILETYPE (FTP client and server) statement" on page 796
- "JESENTRYLIMIT (FTP server) statement" on page 802
- "JESINTERFACELEVEL (FTP server) statement"
- "JESLRECL (FTP server) statement" on page 806
- "JESRECFM (FTP server) statement" on page 808

JESINTERFACELEVEL (FTP server) statement

Use the JESINTERFACELEVEL statement to specify the FTP-to-JES interface to be used by the installation. With JESINTERFACELEVEL 1, FTP users can submit jobs to JES, retrieve held output matching their logged-in user ID plus one character, and delete held jobs matching their logged-in user ID plus one character.

With JESINTERFACELEVEL 2, FTP users can retrieve and delete any job in the system for which they have the security access facility (SAF) resource class JESSPOOL access. Their ability to submit jobs is governed by the JESJOBS class SAF resource. JESINTERFACELEVEL 2 should only be specified if security measures are in place to ensure process access to JES output. For more information about SDSF security see z/OS SDSF Operation and Customization.

JESINTERFACELEVEL 2 uses the SAPI interface to JES, so READ authority to the JESSPOOL resource is required to list job status or retrieve job output. See z/OS JES2 Initialization and Tuning Guide for more information about JES security. See z/OS MVS Using the Subsystem Interface for more information about the SAPI interface.

The SAF controls used for JESINTERFACELEVEL 2 are essentially a subset of those used by SDSF. Therefore, if an installation has customized SAF facilities for SDSF, it is configured for FTP JES JESINTERFACELEVEL 2.

JESSPOOL defines resource names as [nodeid].[userid].[jobname].[Dsid].[dsname]. An FTP user can delete job output if it has ALTER access to the resource that matches its node ID, user ID, and job name (generics can be used). If the FTP client has READ access to the resource, it can list or retrieve the job output. FTP uses three filters to control the display of jobs. These filters employ SDSF resources. The first filter, JESSTATUS, can be changed by an FTP client by way of the SITE command to filter jobs in INPUT, ACTIVE, or OUTPUT state. The second filter, JESOWNER, has the value of the logged-in user ID by default. The third filter, JESJOBNAME, has the value of the logged-in user ID plus an asterisk (*) by default. JESSTATUS uses the SDSF resources ISFCMD.DSP.INPUT.jesx, ISFCMD.DSP.ACTIVE.jesx, and ISFCMD.DSP.OUTPUT.jesx. At login time, the default value for JESSTATUS is set to ALL if READ access is allowed to all three classes. Otherwise, the server attempts to set the value to OUTPUT, ACTIVE, and then INPUT if the appropriate READ access is allowed. If no READ access is allowed to any of the classes, JESSTATUS is set to OUTPUT but JESOWNER and JESJOBNAME cannot be changed from their default values. In this way, SAF controls can be put in place to limit FTP users to whatever status of jobs an installation requires.

Authority to change JESOWNER is obtained by way of READ access to RACF profile ISFCMD.FILTER.OWNER. Authority to change JESJOBNAME is obtained by way of of READ access to RACF profile ISFCMD.FILTER. An FTP client with READ access to ISFCMD.FILTER.OWNER is allowed to change the JESOWNER parameter by way of the SITE command. An FTP client with READ access to ISFCMD.FILTER.PREFIX is allowed to change the JESJOBNAME parameter by way of the SITE command.

Syntax

```
     ┌─JESINTERFACELevel  1──┐
►►───┼───────────────────────┼──────────────────────────────────────────────►◄
     ├─JESINTERFACELlevel  1─┤
     └─JESINTERFACELlevel  2─┘
```

Parameters

1 Specifies that FTP users can submit jobs to JES, retrieve held output matching their logged-in user ID plus one character, and delete held jobs matching their logged-in user ID plus one character. This is the default.

2 Specifies that FTP users can retrieve and delete any job in the system for which they have the security access facility (SAF) resource class JESSPOOL access. Their ability to submit jobs is governed by the JESJOBS class SAF resource.

Guideline: JESINTERFACELEVEL 2 should only be specified if security measures are in place to ensure process access to JES output.

Examples

The following code is an example of commands used to allow all FTP users other than USER1 the ability to change JESOWNER. USER1 is only allowed the default JESOWNER value and not allowed to change JESOWNER by way of the SITE command.

```
JESOWNER: setropts classact(SDSF) refresh
rdefine SDSF (isfcmd.filter.owner) uacc(read)
permit isfcmd.filter.owner access(none) class(SDSF) id(user1)
setropts classact(SDSF) refresh
```

Requirement: If JESINTERFACELEVEL 2 is specified, an installation must ensure that security measures are in place to control FTP client access to jobs.

Result: This statement applies only when FILETYPE JES is active.

Related topics

- "FILETYPE (FTP client and server) statement" on page 796
- "JESENTRYLIMIT (FTP server) statement" on page 802
- "JESGETBYDSN (FTP server) statement" on page 803
- z/OS Communications Server: IP Configuration Guide for information about JESINTERFACELEVEL

JESLRECL (FTP server) statement

Use the JESLRECL statement to specify the record length of the jobs being submitted. You can also use the SITE command to set this keyword.

Syntax

```
       ┌─JESLRECL 80──────────┐
►►─────┤                      ├────────────────────────────────────────►◄
       └─JESLRECL──length─────┘
```

Parameters

length
  The record length of the job being submitted. The valid range is 1 - 254. The default is 80. If you specify *length* as *, FTP uses the length value from the LRECL statement.

Examples

Explicitly set the logical record length for JES jobs to 80:

```
JESLRECL 80
```

Usage notes

- If JESLRECL * is specified, the LRECL value is used for jobs being submitted.
- This statement applies only when FILETYPE JES is active.

Related topics

- "FILETYPE (FTP client and server) statement" on page 796
- "JESGETBYDSN (FTP server) statement" on page 803

• “LRECL (FTP client and server) statement” on page 813

JESPUTGETTO (FTP server) statement

Use the JESPUTGETTO statement to specify the number of seconds of the JES PutGet timeout.

The JES PutGet timeout is used when the FTP client performs a GET with a source and a target name. The source job is submitted to JES. The server waits until the JES PutGet timeout expires or until the job completes. If the job completes, it stores the output in the target name file. If the job does not complete, the FTP client displays the server reply to the end user.

Syntax

```
     ┌─JESPUTGETTO 600─────────┐
►►───┼─────────────────────────┼──────────────────────────────────────────────►◄
     └─JESPUTGETTO──seconds────┘
```

Parameters

seconds
: The number of seconds of the JES PutGet timeout. The valid range is 0 - 86 400 (24 hours). The default is 600 (10 minutes).

Examples

Set the number of seconds of the JES PutGet timeout to 300:

```
JESPUTGETTO 300
```

Usage notes

- The JESPUTGETTO value should be high enough for most jobs to complete within the specified time but not be so high (for example, 86400) that end users wait excessive amounts of time for job completion.
- Use 86400 if the JES PutGet is done *only* from batch jobs that must wait for the job to complete and end user wait time is not an issue.
- This statement applies only when FILETYPE JES is active.

Related topics

- “ANONYMOUSFILETYPEJES (FTP server) statement” on page 742
- “FILETYPE (FTP client and server) statement” on page 796
- “JESENTRYLIMIT (FTP server) statement” on page 802
- “JESGETBYDSN (FTP server) statement” on page 803
- “JESINTERFACELEVEL (FTP server) statement” on page 804
- “JESLRECL (FTP server) statement” on page 806
- “JESRECFM (FTP server) statement” on page 808

JESRECFM (FTP server) statement

Use the JESRECFM statement to specify the record format of jobs being submitted. This is the record format used during dynamic allocation of the internal reader when submitting jobs to JES. You can also use the SITE command to set this keyword.

Syntax

```
     ┌─JESRECFM F──────────┐
►►───┼─────────────────────┼──────────────────────────────────────────────────►◄
     └─JESRECFM───┬─F─┬────┘
                  ├─V─┤
                  └─*─┘
```

Parameters

F Fixed record length. This is the default.

V Variable record format.

***** Uses the record format specified on the RECFM statement.

Examples

Use fixed record format:

```
JESRECFM F
```

Usage notes

- Use only the value F when running on JES2 systems.
- If FTP cannot allocate the internal reader, the FTP client receives a `550 JES internal reader allocation failed` reply when submitting jobs to JES.
- This statement applies only when FILETYPE JES is active.

Related topics

- "FILETYPE (FTP client and server) statement" on page 796
- "JESGETBYDSN (FTP server) statement" on page 803
- "RECFM (FTP client and server) statement" on page 833

KEYRING (FTP client and server) statement

Use the KEYRING statement to define the key ring that contains the certificate to be used during the TLS handshake.

Server Specifies the key ring database on the server's system.

Client Specifies the key ring database on the client's system.

Syntax

```
►►───┬─────────────────────────────────────────┬──────────────────────────────►◄
     └─KEYRING───┬─keyringname─────────┬───────┘
                 └─userid/keyringname──┘
```

Parameters

userid/keyringname
: Allows multiple FTP users to share one key ring owned by another user. The keyringname value is the SAF key ring created by using the RACF ADDRING function.

Restrictions:

- The userid value must be the user that actually owns the key ring.
- All users must have READ and UPDATE access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class when using an SAF key ring owned by another user.

Examples

```
KEYRING /u/user33/keyring/key.kdb
```

```
KEYRING user33/ftpring
```

```
KEYRING ftpring
```

Guideline: If the *userid* is omitted, the current user ID is used.

Usage notes

- KEYRING is required if TLS is used as a security mechanism.
- The EXTENSIONS AUTH_TLS statement must be coded for this statement to be used by the FTP server.
- The SECURE_MECHANISM TLS statement must be coded for this statement to be used by an FTP client.

Related topics

- "EXTENSIONS (FTP client and server) statement" on page 792
- "SECURE_MECHANISM (FTP client) statement" on page 854
- "TLSMECHANISM (FTP client and server) statement" on page 879
- See z/OS Communications Server: IP Configuration Guide for more information about SSL/TLS security, key rings, and certificates and SSL/TLS.

LISTLEVEL (FTP server) statement

Use the LISTLEVEL statement to specify the format of the LIST reply.

Syntax

```
   ┌─LISTLEVEL 0──────┐
►►─┤                  ├──────────────────────────────────────────────►◄
   └─LISTLEVEL───┬─0──┘
                 └─1─┘
```

Parameters

0
: Specifies that PDS, PDSE and z/OS UNIX data sets are displayed with a DSORG value of PO.

1
: Specifies that PDS data sets are displayed with a DSORG value of PO, PDSE data sets are displayed with a DSORG value of PO-E, and z/OS UNIX data sets are displayed with a DSORG value of z/OS UNIX file system.

Examples

Set the LISTLEVEL parameter value to 1 when you want to distinguish PDS, PDSE and z/OS UNIX data sets in the LIST reply:

```
LISTLEVEL 1
```

LISTSUBDIR (FTP client and server) statement

Use the LISTSUBDIR statement to indicate whether wildcard searches should span subdirectories or apply only to the current working directory. You can use the SIte and LOCSIte subcommands to reset this keyword.

Server This setting applies when processing the NLST command. The z/OS FTP client sends an NLST command to the server when issuing any of the following subcommands:

- LS *
- MDELETE *
- MGET *

Client This setting applies when the z/OS FTP client issues an MPUT * subcommand.

This statement only applies when the asterisk (*) wildcard symbol is used in the filename parameter and the GLOB subcommand is set to expand metacharacters in file names. The ls, mdelete, mget and mput subcommands search only the subdirectories of the current path. They do not search multiple depths of subdirectories.

Syntax

```
   ┌─LISTSUBDIR TRUE──────────┐
►►─┼──────────────────────────┼──────────────────────────────────────────────►◄
   └─LISTSUBDIR──┬─TRUE──┬────┘
                 └─FALSE─┘
```

Parameters

TRUE
This is the default. Indicates the files in the subdirectories of the current working directory are listed when processing wildcard searches.

FALSE
Indicates that the files in the subdirectories of the current working directory are not listed when processing wildcard searches.

Examples

Directory /u/user1/xx contains the following files and subdirectory:

```
areadme (file)
file_xx (file)
readme_xx (file)
ggg (subdirectory)
```

Directory /u/user1/xx/ggg contains the following files and subdirectory:

```
file_ggg (file)
zzz (subdirectory)
```

Directory /u/user1/xx/ggg/zzz contains the following files and subdirectory:

```
file_zzz (file)
rrr (subdirectory)
```

The following display shows these files and directories:

```
250 HFS directory /u/user1/xx is the current working directory
ftp> ls - l
200 Port request OK.
125 List started OK
total 40
-rwx------      1       IBMUSER    0                   48 Oct 29  21:14 areadme
-rwx------      1       IBMUSER    0                   10 Nov  1  16:02 file_xx
drwxrwxrwx      3       IBMUSER    0                 8192 Nov  1  16:00 ggg
-rwx------      1       IBMUSER    0                   23 Oct 29  21:06 readme_xx
250 List completed successfully.
260 bytes received in 0.03 seconds (8.67 Kbytes/sec)
ftp> cd gg
260 HFS directory /u/user1/xx/ggg is the current working directory
ftp> ls - l
200 Port request OK.
125 List started OK
total 24
-rwx------      1       IBMUSER    0                    6 Oct 29  16:00 file_ggg
drwxr-x---      3       IBMUSER    0                 8192 Nov  1  16:01 zzz
250 List completed successfully.
133 bytes received in 0.02 seconds (6.65 Kbytes/sec)
cd zzz
250 HFS directory /u/user1/xx/ggg/zzz is the current working directory
ftp> ls - l
200 Port request OK.
125 List started OK
total 24
-rwx------      1       IBMUSER    0                    4 Nov 29  16:00 file_zzz
drwxr-xr-x     3       IBMUSER    0                 8192 Nov  1  16:01 rrr
250 List completed successfully.
133 bytes received in 0.01 seconds (13.30 Kbytes/sec)
```

If you have coded LISTSUBDIR FALSE in the server's FTP.DATA file or specified SITE NOLISTSUBDIR, the client sees the following display:

```
257 "/u/user1/xx" is the current working directory
ftp> ls *
200 Port request OK.
125 List started OK
areadme
file_xx
readme_xx
250 List completed successfully.
29 bytes received in 0.02 seconds (1.45 Kbytes/sec)
```

If you have coded LISTSUBDIR TRUE in the server's FTP.DATA file or specified SITE LISTSUBDIR, the client sees the following display:

```
257 "/u/user1/xx" is the HFS working directory
ftp> ls *
200 Port request OK.
125 List started OK
areadme
file_xx
ggg/file_ggg
readme_xx
250 List completed successfully.
42 bytes received in 0.04 seconds (1.05 Kbytes/sec)
```

When spanning subdirectories with the wildcard *, the file ggg/file_ggg is shown. However, the file ggg/file_zzz is not shown because the subdirectory span is only one level deep.

Restriction: The LISTSUBDIR statement applies to z/OS UNIX file operations only. MVS data set operations are not affected.

Related topics

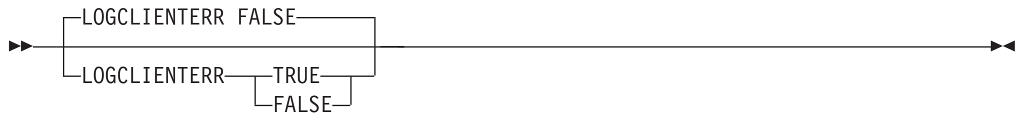
For more information about the following topics, see z/OS Communications Server: IP User's Guide and Commands:
- ls
- mget
- mput
- mdelete
- glob
- site for the LISTSUBDIR option
- locsite for the LISTSUBDIR option

LOGCLIENTERR (FTP client) statement

Use the LOGCLIENTERR statement to specify whether the FTP client should log client errors with message EZZ9830I.

Result: Message EZZ9830I is issued for any error that causes FTP to exit. If you have not configured FTP to exit on error by specifying the EXIT or EXIT=*nn* parameter on the FTP command, or by coding the CLIENTEXIT TRUE statement in FTP.DATA, the FTP client will issue message EZZ9830I on the first error that would have caused FTP to exit if you configured FTP to exit on error.

Syntax



Parameters

TRUE
: Specifies that the FTP client should log message EZZ9830I when an FTP client subcommand fails.

FALSE
: Specifies that the FTP client should not log message EZZ9830I when an FTP client subcommand fails. This is the default.

Examples

```
LOGCLIENTERR  TRUE      ; log client errors
```

Related topics

- “CLIENTERRCODES (FTP client) statement” on page 767

- FTP client error logging in z/OS Communications Server: IP User's Guide and Commands

LOGINMSG (FTP server) statement

Use the LOGINMSG statement to specify the file containing messages to be displayed to FTP users when they have successfully logged in. This statement affects only named FTP clients as opposed to FTP clients logged in as anonymous.

Syntax

```
►►──LOGINMSG──file-path──────────────────────────────────────────►◄
```

Parameters

file-path
The fully qualified z/OS UNIX pathname or the fully qualified MVS data set name of the file whose contents are displayed whenever a user logs in to FTP.

Requirements:
- A z/OS UNIX pathname must start with a slash (/).
- An MVS data set must not start with a slash character.

Examples

Use the following statement if the FTP login message is kept in the file /etc/ftp.login:

```
LOGINMSG /etc/ftp.login
; Welcome message for FTP users
```

Usage notes

- LOGINMSG does not apply to anonymous user logins. To provide this function to anonymous users, use the ANONYMOUSLOGINMSG statement.
- When a known FTP user successfully logs in, the FTP server searches for the file specified by file-path. The contents of the file are returned to the FTP user as 230-prefixed replies. If the file specified by file-path does not exist, no messages are returned to the FTP client and no login messages are displayed to the end user.

Related topics

- "ANONYMOUSLOGINMSG (FTP server) statement" on page 750
- "BANNER (FTP server) statement" on page 756
- "HFSINFO (FTP server) statement" on page 799
- "MVSINFO (FTP server) statement" on page 820

LRECL (FTP client and server) statement

Use the LRECL statement to specify the size of the logical records in a data set.

Server This setting applies when creating files on the server's system. For example, with a PUT subcommand.

Client This setting applies when creating files on the client's system. For example, with a GET subcommand.

Syntax

```
            ┌─256────┐
►►──LRECL───┼────────┼──────────────────────────────────────────────►◄
            └─length─┘
```

Parameters

length
The size of the records in a data set. The valid range is 0 - 32 760 or *x*. The default is 256. *x* corresponds to a length of 32 768.

Examples

Set the logical record length to 128 bytes:

```
LRECL 128
```

Specify no value for LRECL to allow the LRECL of a model DCB data set or SMS dataclass to be used:

```
LRECL
```

Usage notes

- The record size attribute can be obtained from an SMS data class using the DATACLASS configuration statement, from a model data set using the DCBDSN configuration statement, or from the LRECL statement.
- Use LRECL without a size if you have:
  - Specified a DATACLASS statement and want to use the record size from the data class, or
  - Specified a DCBDSN statement and want to use the record size from the model data set.
- If you specify a DATACLASS, a DCBDSN, and LRECL without size, the value from the model data set is used.
- To override the record size attribute from the DATACLASS or DCBDSN settings:
  - Specify LRECL with a value other than 0, or
  - Do not specify the LRECL statement, and use the default.
- If you specify no value for *length*, FTP does not specify the size of the records in a data set.

Related topics

- "DATACLASS (FTP client and server) statement" on page 770
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- "DCBDSN (FTP client and server) statement" on page 776
- "JESLRECL (FTP server) statement" on page 806
- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.

MBDATACONN (FTP client and server) statement

Use the MBDATACONN statement to define the conversions between a file system code page and a network transfer code page during data transfer. This statement affects the conversion of double-byte character set (DBCS) and multibyte character set (MBCS) data and is used when the ENCODING MBCS statement is coded. You can also use the SIte and LOCSIte subcommands to set this keyword and to set ENCODING to MBCS.

Server Specifies the multibyte code pages used by the server.

Client Specifies the multibyte code pages used by the client.

Syntax

```
►►──MBDATACONN──(file_system_codepage,network_transfer_codepage)──────────────►◄
```

Parameters

file_system_codepage
Specifies the name of the file system code page.

network_transfer_codepage
Specifies the network transfer code page.

Examples

To code MBDATACONN:

```
MBDATACONN    (IBM-1388,IBM-5488)
```

Usage notes

MBDATACONN is in effect only when ENCODING has a value of MBCS.

Table 48 shows the supported code page pairs.

Table 48. Supported code page pairs

Support for:	file_system_codepage	network_transfer_codepage
Chinese standard GB18030	IBM-1388 or UTF-8	IBM-5488
BIG5	IBM-937	IBM-950 or BIG5
EUCKANJI	IBM-930	IBM-eucJP
JIS78KJ (JISROMAN	IBM-930	IBM-5053
JIS78KJ (ASCII	IBM-939	IBM-5055
JIS83KJ (JISROMAN	IBM-930	IBM-5052
JIS83KJ (ASCII	IBM-939	IBM-5054
KSC5601	IBM-933	IBM-949
SCHINESE	IBM-935	IBM-1381
SJISKANJI	SJISKANJI IBM-930 or IBM-939	IBM-932 or IBM-eucJC
TCHINESE	IBM-937	IBM-948
UNICODE file transfer	UTF-8, UTF-16	UTF-8, UTF-16, UTF-16BE, UTF-16LE

Other code page pairs might be accepted when specified. However, the ones listed in Table 48 on page 815 have been verified to produce the support that is listed in the table.

Related topics

- "ENCODING (FTP client and server) statement" on page 790
- "MBSENDEOL statement (FTP client and server) statement" on page 817
- "MBREQUIRELASTEOL (FTP client and server) statement"
- "SBDATACONN (FTP client and server) statement" on page 840
- Table 20 on page 450

MBREQUIRELASTEOL (FTP client and server) statement

Use the MBREQUIRELASTEOL statement to specify whether FTP requires the last record of incoming multibyte files to end with the FTP standard EOL sequence.

Server This setting applies when the server is receiving a multibyte file from the client.

Client This setting applies when the client is receiving a multibyte file from the server.

Syntax

```
   ┌─MBREQUIRELASTEOL TRUE────────┐
▶▶─┤                              ├──────────────────────────────────────▶◀
   └─MBREQUIRELASTEOL─────────────┘
                            ├─TRUE──┤
                            └─FALSE─┘
```

Parameters

TRUE
FTP reports an error when a multibyte file is received from the network without an EOL sequence in the last record received and aborts the file transfer. The CONDDISP configuration option determines whether the file or data set is saved or deleted.

FALSE
FTP does not report an error when a multibyte file is received from the network without an EOL sequence in the last record received. The file or data set is stored.

Results:

- If MBREQUIRELASTEOL is set to FALSE, and you have coded CHKCONFIDENCE TRUE in FTP.DATA, the confidence level reported when a multibyte file is received from the network without an EOL sequence in the last record is High.
- If MBREQUIRELASTEOL is set to TRUE, and you have coded CHKCONFIDENCE TRUE in FTP.DATA, the confidence level reported when a multibyte file is received from the network without an EOL sequence in the last record is Low.

Examples

To enable the FTP server to receive multibyte files that are sent with no EOL sequence on the final record, code the following statement in the server's FTP.DATA:

```
MBREQUIRELASTEOL FALSE
```

Related topics

- “ENCODING (FTP client and server) statement” on page 790
- “MBDATACONN (FTP client and server) statement” on page 815

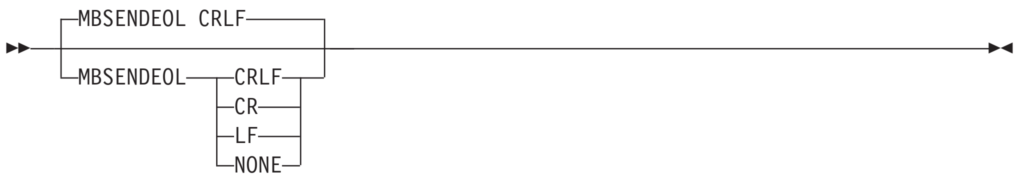
MBSENDEOL statement (FTP client and server) statement

Use the MBSENDEOL statement to tell the FTP client or server what end-of-line (EOL) sequence to use when ENCODING is MBCS, Type is ASCII, Mode is Stream, and file transfer is outbound. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server When ENCODING is MBCS, data Type is ASCII, Mode is Stream, and files are sent from the server, this statement instructs the server which EOL sequence to append to each line of text.

Client When ENCODING is MBCS, data type is ASCII, Mode is Stream, and files are sent from the client, this statement instructs the client which EOL sequence to append to each line of text.

Syntax



Parameters

CRLF
When translating multi-byte data to ASCII, append a carriage return (x'0D') and line feed (x'0A') to each line of text. This is the default and the standard EOL sequence defined by RFC 959. The z/OS server and client can receive ASCII data in this format only.

CR When translating multi-byte data to ASCII, append only a carriage return (x'0D') to each line of text.

LF When translating multi-byte data to ASCII, append only a line feed (x'0A') to each line of text.

NONE
When translating multi-byte data to ASCII, append no EOL sequence.

Results:

- This statement applies only to the end-of-line sequence used on the data connection. The control connection end-of-line sequence is not affected.

- SBCS, DBCS, and UCS-2 translations are not affected by this setting. UTF-8 and UTF-16 translations are affected by this setting

Rule: The MBSENDEOL setting CRLF is appropriate for most file transfers. Do not use an alternate MBSENDEOL setting unless you have verified that the recipient FTP can handle the alternate value.

Client Do not code an alternate MBSENDEOL value if your server is a z/OS FTP server. The z/OS FTP server does not support alternate MBSENDEOL values for inbound file transfer.

Server Do not code an alternate MBSENDEOL value if your client is a z/OS FTP client. The z/OS FTP client does not support alternate MBSENDEOL values for inbound file transfer.

Examples

Use LF as the EOL sequence when ENCODING is MBCS, Type is ASCII, and data transfer is outbound. Code as follows:

```
MBSENDEOL  LF
```

Related topics

- "ENCODING (FTP client and server) statement" on page 790
- "MBDATACONN (FTP client and server) statement" on page 815
- "SBSENDEOL statement (FTP client and server) statement" on page 841

MGMTCLASS (FTP client and server) statement

Use the MGMTCLASS statement to specify the SMS management class to be assigned to newly allocated data sets.

Server This setting applies when creating files on the server's system.

Client This setting applies when creating files on the client's system.

One of the attributes obtained from the management class is the retention period setting. If you specify a management class, then the retention period is obtained from the management class. The value of the management class's retention period can be overridden.

- If a data class (DATACLASS) is specified, the retention period in the data class can override it.
- If a model data set (DCBDSN) is specified, its retention period overrides both the data class value and the management class value.
- If you specify a value for RETPD statement, the value you specify overrides any data class setting, model data set value and any management class setting.

However, regardless of where the retention period value is obtained, when attempting to override the value set in the management class, the actual resulting retention period setting depends on the retention period limit defined in the management class. A management class is defined with a retention limit value as well as a retention period. If you attempt to override the management class's retention period, the override value must be within the retention period limit defined in the management class. Otherwise, the retention period used is the management class's retention limit value.

Syntax

```
►►──MGMTCLASS──class──────────────────────────────────────────────►◄
```

Parameters

class
    The SMS management class.

Examples

Set the SMS management class for new data sets to TCPMGMT:

```
MGMTCLASS TCPMGMT
```

Related topics

- "DATACLASS (FTP client and server) statement" on page 770
- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "STORCLASS (FTP client and server) statement" on page 877

MIGRATEVOL (FTP client and server) statement

Use the MIGRATEVOL statement to specify the volume ID for migrated data sets under the control of a storage management system other than HSM.

Server This setting applies when accessing files on the server's system.

Client This setting applies when accessing files on the client's system.

Syntax

```
     ┌─MIGRATEVOL MIGRAT──────┐
►►───┼────────────────────────┼───────────────────────────────────►◄
     └─MIGRATEVOL──volume_id──┘
```

Parameters

volume_id
    The volume ID for migrated data sets. The default volume ID is MIGRAT.

Examples

Set the volume ID for migrated data sets to MIGRIX:

```
MIGRATEVOL MIGRIX
```

Related topics

"RETPD (FTP client and server) statement" on page 838

MVSINFO (FTP server) statement

Use the MVSINFO statement to specify MVS data sets whose contents are displayed to the user when the user changes directories. The statement identifies a low-level qualifier (LLQ) that is appended to the current path whenever an FTP user changes directories to an MVS data set.

Syntax

```
►►──MVSINFO──MVS-LLQ──────────────────────────────────────────────►◄
```

Parameters

MVS-LLQ
: The MVS-LLQ is the MVS low-level qualifier (LLQ) appended to the current MVS path whenever an FTP client changes directories to an MVS data set. If a data set matches the current path appended LLQ, the contents of the data set are returned to the FTP client and displayed to the end user.

Examples

To display a readme file the first time a user changes directory to high-level qualifiers, use the following statement. In this example, an MVS high-level qualifier of `productname` might have a readme file for each product, and any time a user changed directory to the `productname`, the readme file would be displayed.

```
MVSINFO README
```

Usage notes

MVSINFO does not apply to anonymous users. Use the ANONYMOUSMVSINFO statement to define the informational banner used for anonymous users.

Related topics

- "ANONYMOUSHFSINFO (FTP server) statement" on page 747
- "ANONYMOUSMVSINFO (FTP server) statement" on page 751
- "BANNER (FTP server) statement" on page 756
- "HFSINFO (FTP server) statement" on page 799
- "LOGINMSG (FTP server) statement" on page 813

MVSURLKEY (FTP server) statement

Use the MVSURLKEY statement to specify a token that users can enter as part of an FTP URL to encode an MVS data set name.

Syntax

```
►►──MVSURLKEY──key────────────────────────────────────────────────►◄
```

Parameters

key
: An arbitrary token users can enter in an FTP URL to signify that an MVS data

set follows. Although the FTP server accepts any value, avoid symbols FTP clients might interpret as special characters or meta characters. For example, the # character is acceptable to the FTP server, but some Web browsers use the # character as a special character.

Examples

Use the following example to permit users to enter MVSDS in an FTP URL in order to tell the FTP server an MVS data set name follows:

```
MVSURLKEY MVSDS
; code this in FTP.DATA
```

Code the following as an FTP URL to indicate that 'USER1.PROCLIB(FTPD)' is an MVS data set, not a z/OS UNIX data set:

```
ftp://user1@mvs098.tcp.raleigh.ibm/MVSDS/'user1.proclib(ftpd)';type=a
```

Usage notes

- The key specified for MVSURLKEY can be set to be the same key used for the Websphere server to designate FTP URL encodings.
- The FTP server accepts an arbitrary string. Avoid characters the FTP client might interpret as metacharacters or special characters.

MYOPENTIME (FTP client) statement

Use the MYOPENTIME statement to specify the amount of time the FTP client waits for a session to open before terminating the attempt and reporting an error.

Syntax

```
      ┌─MYOPENTIME 120──────┐
►►────┼─────────────────────┼──────────────────────────────────────────►◄
      └─MYOPENTIME──seconds─┘
```

Parameters

seconds
: The number of seconds to which the timer is set. The valid range is 0 (MYOPENTIME not used) or 15-86 400. The default is 60 seconds.

Examples

```
MYOPENTIME  60  ; wait 60 seconds
```

Related topics

See the FTP command and the FTP environment information in z/OS Communications Server: IP User's Guide and Commands.

NETRCLEVEL (FTP client) statement

Use the NETRCLEVEL statement to specify how the FTP client searches the NETRC data set for FTP server hostnames. This statement applies only if you have defined a NETRC data set for the client to use.

Syntax

```
    ┌─NETRCLEVEL 1─────┐
►►──┤                  ├─────────────────────────────────────────────────►◄
    └─NETRCLEVEL──┬─1─┬┘
                  └─2─┘
```

Parameters

1 The FTP client searches the NETRC data set for the hostname as it was entered by the user: IP address or DNS name. If the client is running in batch mode, the client looks for the hostname in the NETRC data set only if a NETRC DD card is part of the batch job. This is the way the FTP client processed server hostnames up to and including release 320. This is the default.

2 The FTP client searches the NETRC data set for the hostname as it was entered by the user if the user entered a DNS name, or an IP address that cannot be resolved to a DNS name. If the hostname is an IP address that resolves to a DNS name, the FTP client searches the NETRC data set for the DNS name. If the FTP client is running as a batch job and no NETRC DD card is included, the FTP client uses 'userid.NETRC' as the NETRC data set.

Examples

```
NETRCLEVEL  2 ; convert IP addresses
```

Usage notes

The FTP server hostname is the DNS name or IP address the user entered to log in to the FTP server. This statement applies only if the FTP client is using a NETRC data set or file.

Related topics

See z/OS Communications Server: IP User's Guide and Commands for information about how to use the NETRC data set during the login process.

NONSWAPD (FTP server) statement

Use the NONSWAPD statement to allow the FTP daemon address space to run with nonswappable memory.

Syntax

```
    ┌─NONSWAPD FALSE─────┐
►►──┤                    ├───────────────────────────────────────────────►◄
    └─NONSWAPD──┬─FALSE─┬┘
                └─TRUE──┘
```

Parameters

False
Do not set daemon nonswappable. This is the default.

True
Set daemon nonswappable.

Examples

To request that the daemon address space be set nonswappable, code:

```
NONSWAPD TRUE
```

Usage notes

- The FTP daemon must have at least READ access to the FACILITY class resource BPX.STOR.SWAP to enable this option.
- If the call to set the address space nonswappable fails for any reason, the daemon uses swappable memory and continue.
- When an application makes an address space nonswappable, it might cause additional real storage in the system to be converted to preferred storage. Because preferred storage cannot be configured offline, using this option can reduce the installation's ability to reconfigure storage in the future. See z/OS MVS Programming: Resource Recovery for more information.

PASSIVEDATACONN (FTP server) statement

When the server receives a PASV or EPSV command, it opens a listening socket. Any entity can connect to the listening socket. Use the PASSIVEDATACONN statement to direct the server to verify the peer IP address of the data socket is the client's IP address.

Syntax

```
     ┌─PASSIVEDATACONN UNRESTRICTED─────┐
►►───┤                                  ├──────────────────────────────────►◄
     └─PASSIVEDATACONN───┬─UNRESTRICTED─┤
                         └─NOREDIRECT───┘
```

Parameters

UNRESTRICTED
: The server accepts a passive data connection from any IP address. This is the default.

NOREDIRECT
: The server verifies the peer address of the data socket is the client's IP address. If it is not, the server closes the data socket.

Guideline: The server cannot be the passive server in a three way (proxy) data transfer when NOREDIRECT is coded, because the server rejects an attempt by the active server to connect to its passive socket.

Examples

Use the following example to set the server to reject passive data connections with IP address different from the IP addresses of the control connections:

```
PASSIVEDATACONN NOREDIRECT
PASSIVEDATACONN N
```

PASSIVEDATAPORTS (FTP server) statement

Use the PASSIVEDATAPORTS statement to assign a range of port numbers for the FTP server to use as listening data socket ports.

Syntax

```
►►──PASSIVEDATAPORTS (low_port, high_port)──────────────────────────────────►◄
```

Parameters

low_port
The lowest port number the FTP server is allowed to use when creating a listening data socket. The lowest number allowed for low_port is 1 024.

high_port
The highest port number the FTP server is allowed to use when creating a listening data socket. The highest number allowed for high_port is 65 535.

By default, the FTP server allows the stack to select a port number from its entire range of ephemeral ports for listening data sockets. PASSIVEDATAPORTS affects ports selected for the data connection only; the control connection ports are not affected. PASSIVEDATAPORTS is useful in conjunction with firewalls that restrict the range of port numbers allowed to FTP.

Guideline: Code a PORTRANGE AUTHPORT statement in PROFILE.TCPIP to reserve the ports you have specified with PASSIVEDATAPORTS. If you are using a sysplex DVIPA to distribute the FTP server workload with sysplex ports, code the same PORTRANGE AUTHPORT statement for each participating stack in the sysplex.

Restriction: If you have PORTRANGE statements in PROFILE.TCPIP that reserve ports for a different application, and those reserved ports intersect with the PASSIVEDATAPORTS ports, the FTP server is never able to obtain those ports.

Examples

To restrict the server's choice of ports for listening data sockets to ports from 50000 to 50099, code the following statement in FTP.DATA:

```
PASSIVEDATAPORTS (50000,50099)
```

To prevent other applications from consuming ports in the range 50 000 - 50 099, code the following statement in PROFILE.TCPIP:

```
PORTRANGE 50000 100 TCP AUTHPORT
```

PASSIVEIGNOREADDR (FTP client) statement

Use the PASSIVEIGNOREADDR statement to direct the FTP client to ignore the IP address returned from the server on the PASV reply on IPv4 sessions. You can also use the subcommand to set this parameter.

Restrictions:

- The FTP server ignores this statement.

- When EPSV4 and PASSIVEIGNOREADDR are TRUE, the client tries the EPSV command first. If the EPSV command does not succeed, and FRIENDLY is TRUE, then the client tries the PASV command. The PASSIVEIGNOREADDR value determines how the FTP client uses the IP address that is returned by the PASV command.

Syntax

```
   ┌─PASSIVEIGNOREADDR FALSE───────┐
►►─┼───────────────────────────────┼──────────────────────────────────────►◄
   └─PASSIVEIGNOREADDR──┬─FALSE─┬──┘
                        └─TRUE──┘
```

Parameters

FALSE
: For passive mode FTP, specifies that the FTP client uses the IP address and port number from the PASV command reply that is returned by the FTP server for the data connection. This is the default value.

TRUE
: For passive mode FTP, specifies that the FTP client uses the port number from the PASV command reply, and the IP address used to log into the FTP server, for the data connection.

Guideline: If your client has trouble establishing a data connection on an IPv4 encrypted session through a NAT firewall, and the FTP server does not support extended passive mode, coding PASSIVEIGNOREADDR TRUE might help.

Requirement: FWFRIENDLY must also be set to TRUE to enable this function.

Examples

To direct the client to ignore the IP address on the FTP server's PASV reply, use the following code:

```
PASSIVEIGNOREADDR TRUE
```

Related topics

- “EPSV4 (FTP client) statement” on page 791
- “FWFRIENDLY (FTP client) statement” on page 799

PASSPHRASE (FTP server) statement

Use the PASSPHRASE statement to indicate whether the FTP server allows an FTP client to log in to FTP with a password phrase.

Syntax

```
   ┌─PASSPHRASE TRUE───────┐
►►─┼───────────────────────┼──────────────────────────────────────────────►◄
   └─PASSPHRASE──┬─FALSE─┬─┘
                 └─TRUE──┘
```

Parameters

TRUE
The FTP server allows an FTP client to log in to FTP with a password phrase. This is the default value.

FALSE
The FTP server does not allow an FTP client to log in to FTP with a password phrase.

Examples

To allow an FTP client to log in to FTP with a password phrase, code the following statement:

```
PASSPHRASE      TRUE
```

Usage notes

When PASSPHRASE FALSE is configured in FTP.DATA of the server, consider the following two things:

- If an FTP client logs in to FTP with a password of a length that is greater than 8 characters, the password is truncated to 8 characters.
- The FTCHKPWD exit parameter at offset +36 points to a buffer that consists of a 2-byte field, which contains zeros, and is followed by 100 blanks.

PDSTYPE (FTP client and server) statement

Use the PDSTYPE statement in the FTP server and client to indicate whether to allocate MVS directories as partitioned data sets or as partitioned data sets extended. You can also use the SIte and LOCSIte subcommands to set this keyword.

This is one of many statements available to control how MVS directories are allocated when the server receives an MKD command with an MVS PDS name argument.

You should specify PDSTYPE without the PDS or PDSE parameters if the DATACLASS statement is specified and the PDS type from the SMS data class is going to be used.

Server This setting applies when creating an MVS directory on the server's system.

Client This setting applies when creating an MVS directory on the client's system.

Syntax

```
►►──PDSTYPE──┬────────┬──────────────────────────────────────────►◄
             ├─PDS────┤
             └─PDSE───┘
```

Parameters

PDS
Allocate MVS directories as partitioned data sets.

PDSE
Allocate MVS directories as partitioned data sets extended.

Note: If neither PDS or PDSE specified, the default is PDSTYPE.

Examples

Set the directory type to partitioned data set extended:

```
PDSTYPE PDSE
```

Specify PDSTYPE with no value to obtain the PDS type from an SMS data class.

Related topics

- “BLKSIZE (FTP client and server) statement” on page 756
- “DATACLASS (FTP client and server) statement” on page 770
- “DIRECTORY (FTP client and server) statement” on page 781
- “DSNTYPE (FTP client and server) statement” on page 783
- “EATTR (FTP client and server) statement” on page 788
- “LRECL (FTP client and server) statement” on page 813
- “PRIMARY (FTP client and server) statement” on page 830
- “RECFM (FTP client and server) statement” on page 833
- “SECONDARY (FTP client and server) statement” on page 845
- “SPACETYPE (FTP client and server) statement” on page 873

PORTCOMMAND (FTP server) statement

Use the PORTCOMMAND statement to specify whether the PORT command is accepted or rejected. If REJECT is coded, this limits the use of commands such as GET, PUT, MPUT, MGET, and APPEND in PROXY mode. If not in PROXY mode, and REJECT is coded, the FTP server uses the same ephemeral port for the data connection that is used for the control connection. When issuing multiple commands that use the data connection, delays can occur.

Syntax

```
   ┌─PORTCOMMAND ACCEPT─────┐
►►─┼────────────────────────┼──────────────────────────────────────►◄
   └─PORTCOMMAND──┬─ACCEPT─┬┘
                  └─REJECT─┘
```

Parameters

ACCEPT
The PORT and EPRT commands are accepted by the server.

REJECT
The PORT and EPRT commands are rejected by the server.

When PORTCOMMAND is set to REJECT, all PORT and EPRT commands are rejected. PORTCOMMANDPORT and PORTCOMMANDIPADDR settings are disregarded.

Examples

Setting the server to reject all PORT and EPRT commands is shown in the following example:

```
PORTCOMMAND REJECT
```

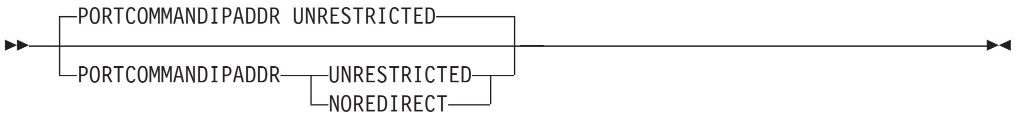
Related topics

- “PORTCOMMANDIPADDR (FTP server) statement”
- “PORTCOMMANDPORT (FTP server) statement”

PORTCOMMANDIPADDR (FTP server) statement

Use the PORTCOMMANDIPADDR statement to direct the server to accept only PORT or EPRT commands whose IP address matches that of the client.

Syntax



Parameters

UNRESTRICTED
: If PORTCOMMAND is set to ACCEPT or unspecified, the server accepts any IP address as a parameter for the PORT and EPRT commands.

NOREDIRECT
: If PORTCOMMAND is set to ACCEPT or unspecified, the server rejects any PORT or EPRT command whose IP address does not match that of the client.

Examples

Setting the server to reject all PORT or EPRT commands with an IP address different from the IP address of the control connection is shown in the following example:

```
PORTCOMMAND ACCEPT
PORTCOMMANDIPADDR NOREDIRECT
```

Usage notes

When PORTCOMMAND is set to REJECT, all PORT and EPRT commands are rejected. PORTCOMMANDPORT and PORTCOMMANDIPADDR settings are disregarded.

Related topics

- “PORTCOMMAND (FTP server) statement” on page 827
- “PORTCOMMANDPORT (FTP server) statement”

PORTCOMMANDPORT (FTP server) statement

Use the PORTCOMMANDPORT statement to specify what range of port values the server accepts as a parameter for the PORT or EPRT command.

Syntax

```
     ┌─PORTCOMMANDPORT UNRESTRICTED───────┐
►►───┤                                    ├──────────────────────────────────────►◄
     └─PORTCOMMANDPORT──┬─UNRESTRICTED─┬──┘
                        └─NOLOWPORTS───┘
```

Parameters

UNRESTRICTED
: If PORTCOMMAND is set to ACCEPT or unspecified, the server accepts any port number as a parameter for the PORT or EPRT command.

NOLOWPORTS
: If PORTCOMMAND is set to ACCEPT or unspecified, the server rejects any PORT or EPRT command specifying a port number lower than 1024.

Examples

Setting the server to reject all PORT or EPRT commands with a port number less than 1024 is shown in the following example:

```
PORTCOMMAND ACCEPT
PORTCOMMANDPORT NOLOWPORTS
```

Usage notes

When PORTCOMMAND is set to REJECT, all PORT and EPRT commands are rejected. PORTCOMMANDPORT and PORTCOMMANDIPADDR settings are disregarded.

Related topics

- "PORTCOMMAND (FTP server) statement" on page 827
- "PORTCOMMANDIPADDR (FTP server) statement" on page 828

PORTOFENTRY4 (FTP server) statement

Use the PORTOFENTRY4 statement to specify the resource profile class name the FTP server should have the USS kernel pass to the security server during login processing for IPv4 clients.

Syntax

```
     ┌─PORTOFENTRY4 TERMINAL──────┐
►►───┤                            ├──────────────────────────────────────────────►◄
     └─PORTOFENTRY4──┬─TERMINAL─┬─┘
                     └─SERVAUTH─┘
```

Parameters

TERMINAL
: The IPv4 client address is always passed as an 8-byte hexadecimal character string resource name in the TERMINAL class.

SERVAUTH
: If the IPv4 client address is mapped into a network security zone by a

NETACCESS statement in the TCPIP PROFILE, the netaccess resource name in the SERVAUTH class is passed. If the client address is not mapped, the TERMINAL class resource name is passed.

Examples

To pass SERVAUTH resource names when mapped, use the following code:

```
PORTOFENTRY4 SERVAUTH
```

Related topics

For more information about network access control and port of entry access control with the FTP server, see z/OS Communications Server: IP Configuration Guide.

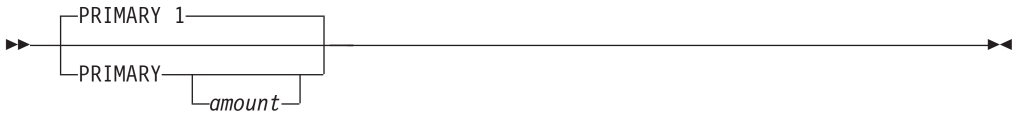
PRIMARY (FTP client and server) statement

Use the PRIMARY statement to specify the number of tracks, blocks, or cylinders (according to SPACETYPE) for primary allocation.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax



Parameters

amount
The number of tracks, blocks, or cylinders. The valid range is 1 - 16 777 215 blocks (the operating system maximum). The default is 1.

- If you specify no value for the *amount* parameter, FTP does not specify the number of tracks, blocks, or cylinders for primary allocation.
- You should specify no value for the *amount* parameter if the DATACLASS statement is specified and the space allocation from the SMS data class is to be used. If the SMS data class is to be used for space allocation, both the PRIMARY and SECONDARY values must be omitted and the value on the SPACETYPE statement is ignored.

  Restriction: If a UNIX file (such as /etc/ftp.data) is being used as the configuration input and no value for the *amount* parameter is specified, the statement should not have any trailing blanks. Ensure that the line ends after the PRIMARY keyword or that a comment is also specified.
- For allocating partitioned data sets, *amount* is the quantity that is allocated for the primary extent.
- For allocating sequential data sets, *amount* is the maximum quantity that is allocated for the primary extent. If a lesser amount is needed to hold the data being transferred, the unused amount is released after the transfer is complete.

Examples

Set the primary allocation to 5 tracks:

```
PRIMARY 5
```

Related topics

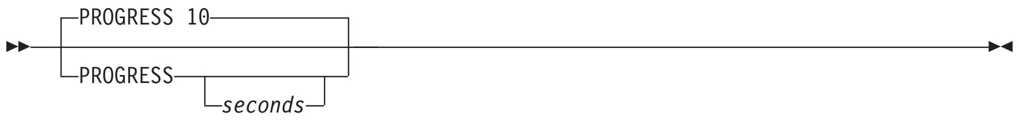
- "DATACLASS (FTP client and server) statement" on page 770
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "SECONDARY (FTP client and server) statement" on page 845
- "SPACETYPE (FTP client and server) statement" on page 873

PROGRESS (FTP client) statement

Use the PROGRESS (FTP client) statement to control the interval between progress report messages generated by the FTP client during an inbound or outbound file transfer.

Client This setting applies when transferring data to or from the FTP client.

Syntax



Parameters

seconds
: Specifies the interval in seconds between progress report messages generated in the FTP client during an inbound or outbound file transfer. Valid values are in the range 10 - 86400, or 0. A value of 0 turns progress reporting off in the FTP client. The default value is 10 seconds. Messages EZA2509I and EZA1485I are generated as part of progress reporting. These messages are generated automatically at 10-second intervals by the FTP client in releases prior to V1R6. Beginning in V1R6, the default behavior is the same as in prior releases, but the length of the interval and whether to generate the messages can be configured by using the PROGRESS parameter setting on the locsite subcommand or by specifying the PROGRESS statement in FTP.DATA.

Examples

To set the progress reporting interval to 30 seconds, use the following code:

```
PROGRESS  30
```

QUOTESOVERRIDE (FTP client and server) statement

Use the QUOTESOVERRIDE statement to indicate the usage of single quotation marks appearing at the beginning of or surrounding a file name.

Server This setting applies to the processing of names by the server.

Client This setting applies to the processing of names by the client.

Syntax

```
   ┌─QUOTESOVERRIDE TRUE──────────┐
►►─┼──────────────────────────────┼──────────────────────────────────────────►◄
   └─QUOTESOVERRIDE──┬─TRUE──┬────┘
                     └─FALSE─┘
```

Parameters

TRUE
: If TRUE is specified, single quotation marks appearing at the beginning and end of a file name are interpreted as meaning the file name contained inside the single quotation marks should override the current working directory instead of being appended to the current working directory. Any single quotation marks inside the beginning and ending quotion mark are treated as part of the file name. This is the default.

FALSE
: If FALSE is specified, a single quote at the beginning of the file name, as well as all other single quotation marks contained in the file name, is treated as part of the actual file name. The entire file name, including the leading single quote, is appended to the current working directory.

Examples

To treat quotation marks as part of file names, enter the following code:

```
QUOTESOVERRIDE FALSE
```

RDW (FTP client and server) statement

Record Descriptor Words (RDWs) are the first 4 bytes at the start of a variable record length file that tell the reading program the actual length of the current record being read. Use the RDW statement to specify whether the RDW from variable format data sets should be retained as data and transmitted or not transmitted.

Server This setting applies when transferring data sets from the server's system.

Client This setting applies when transferring data sets from the client's system.

Syntax

```
   ┌─RDW FALSE──────────┐
►►─┼────────────────────┼────────────────────────────────────────────────────►◄
   └─RDW──┬─TRUE──┬─────┘
          └─FALSE─┘
```

Parameters

TRUE
: Record descriptor words are transferred as data. FTP returns the RDWs as part of the data for variable record format data sets. Depending upon the FTP

implementation, RDWs might not be handled as the user expects. For example, the z/OS Communications Server FTP client might treat received RDWs as as carriage control/line feeds.

FALSE
Record descriptor words are not transferred with the data.

Examples

To specify that a variable record format file's data is transmitted without the descriptors showing the way it was stored on z/OS, use the following code:

```
RDW FALSE
```

Related topics

"RECFM (FTP client and server) statement"

RECFM (FTP client and server) statement

Use the RECFM statement to specify the record format of new, dynamically allocated data sets.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax

```
   ┌─RECFM VB──────────────┐
►►─┼───────────────────────┼──────────────────────────────────────────────►◄
   └─RECFM──┬────────┬─────┘
            └─format─┘
```

Parameters

format
The record format of a data set. Valid record formats are:

- F
- FM
- FA
- FS
- FSA
- FSM
- FB
- FBM
- FBA
- FBS
- FBSM
- FBSA
- V
- VM
- VA

- VS
- VSM
- VSA
- VB
- VBM
- VBA
- VBS
- VBSA
- VBSM
- U
- UA
- UM

The default record format is VB. The meanings of the record formats are:

Format	**Description**
A	Records contain ISO/ANSI control.
B	Blocked records.
F	Fixed record length.
M	Records contain machine code control characters.
S	Spanned records (if variable) or Standard (if fixed).
U	Undefined record length.
V	Variable record length characters.

Examples

Use fixed blocked record format:

```
RECFM FB
```

Specify RECFM with no value to allow the RECFM value of a DCB data set or an SMS dataclass to be used:

```
RECFM
```

Usage notes

- If you specify no value for *format*, no record format is specified when allocating new data sets.
- The record format attribute can be obtained from an SMS data class using the DATACLASS statement, from a model data set using the DCBDSN statement, or from the RECFM statement.
- You should specify no value for *format* if you:
  - Specify the DATACLASS statement and the record format from the SMS data class is to be used, or
  - Specify the DCBDSN and the record format from the model data set is to be used.
- If you specify both a DATACLASS and a DCBDSN, and you specify RECFM with no value, the record format attribute is obtained from the model data set.
- You can override the record format attribute from the DATACLASS or DCBDSN settings by specifying RECFM with a value, or by not specifying the RECFM statement and taking the default.

Related topics

- "DATACLASS (FTP client and server) statement" on page 770
- "DCBDSN (FTP client and server) statement" on page 776
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.

REMOVEINBEOF (FTP client and server) statement

Use the REMOVEINBEOF statement to specify whether the z/OS UNIX EOF (x'1A') is removed from inbound data before the data is stored.

This setting applies to type ASCII inbound file transfers when the data is stored into an MVS sequential data set.

Server This setting applies when the server is the receiving site.

Client This setting applies when the client is the receiving site.

Syntax

```
   ┌─REMOVEINBEOF FALSE─────────┐
►►─┼────────────────────────────┼──────────────────────────────────────────────►◄
   └─REMOVEINBEOF──┬─TRUE──┬────┘
                   └─FALSE─┘
```

Parameters

TRUE
: Specifies that if the inbound data contains a z/OS UNIX EOF (x'1A') as the final byte, it is removed from the data.

FALSE
: Specifies that if the inbound data contains a z/OS UNIX EOF (x'1A') as the final byte, it is not removed from the data.

Examples

Remove the UNIX EOF from the inbound data:

```
REMOVEINBEOF TRUE
```

REPLY226 (FTP server) statement

Use the REPLY226 statement to direct the FTP server to reply to the FTP client with reply code 226 instead of reply code 250 to command sequences described in RFC 959; these command sequences enable the server to choose between reply code 226 and reply code 250.

Tips:

- FTP reply codes are described in RFC 959.
- Generally, reply code 226 or 250 is used after a successful file transfer, after LIST commands, and after NLST commands.

- Reply code 250 (but not 226) is used for a broader class of FTP commands, such as RNTO, DELE, MKD, RMD, CWD.
- RFC 959 describes the command sequences where a server is allowed to reply with either reply code 226 or reply code 250.

Syntax

```
       ┌─REPLY226 FALSE────────┐
▶▶─────┼───────────────────────┼──────────────────────────────────────────────▶◀
       └─REPLY226─────┬─FALSE─┬┘
                      └─TRUE──┘
```

Parameters

FALSE
: Directs the server to reply to the client with code 250 after successful file transfer, and after other FTP commands that enable the server to choose between reply code 250 and reply code 226. This is the default.

TRUE
: Directs the server to reply to the client with reply code 226 instead of reply code 250 after successful file transfer, and after other FTP commands that enable the server to choose between reply code 250 and reply code 226.

Restriction: A server is not always permitted to select reply 226 instead of reply 250. The REPLY226 setting does not override RFC 959 in these cases. For example, RFC 959 stipulates the server must reply with reply code 250 to RMD (remove directory); the REPLY226 setting does not affect the reply code selected for RMD commands.

Examples

To direct the client to reply with code 226 instead of code 250 for successful file transfer, and for other command sequences described in RFC 959 that enable the server to choose between reply code 226 and reply code 250, enter the following code in the server's FTP.DATA:

```
REPLY226 TRUE
```

REPLYSECURITYLEVEL (FTP server) statement

Use the REPLYSECURITYLEVEL statement to specify whether or not to include secure information, such as IP addresses and port numbers, in FTP replies.

Syntax

```
       ┌─REPLYSECURITYLEVEL 0───────┐
▶▶─────┼────────────────────────────┼─────────────────────────────────────────▶◀
       └──┬─REPLYSECURITYLEVEL 0─┬──┘
          └─REPLYSECURITYLEVEL 1─┘
```

Parameters

REPLYSECURITYLEVEL 0
: No restrictions are placed on information included in server FTP replies. This is the default.

REPLYSECURITYLEVEL 1
: No IP addresses, hostnames, port numbers, or server operating system level information is included in FTP replies.

Examples

Direct the server not to divulge secure information such as IP addresses and port numbers in replies to the client:

```
REPLYSECURITYLEVEL 1
```

Usage notes

Suppressing sensitive information such as IP addresses from client replies increases the security of your site; however, such information can be useful for debugging. An alternative to getting this information from server replies is to activate the server trace to capture this information. See z/OS Communications Server: IP Diagnosis Guide for information about diagnosing problems with server traces.

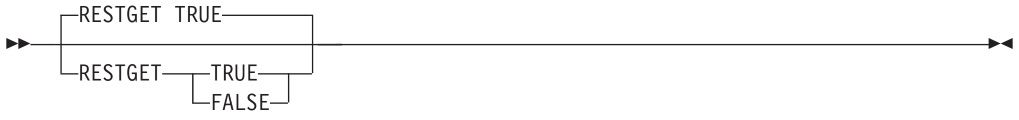
Related topics

"DEBUG (FTP client and server) statement" on page 778

RESTGET (FTP client) statement

Use the RESTGET statement to specify whether the FTP client should open the checkpoint data set for a GET request.

Syntax



Parameters

TRUE
: Specifies that the checkpoint data set is opened for a GET request. This is the default.

FALSE
: Specifies that the checkpoint data set is not opened for a GET request.

Examples

```
RESTGET  FALSE ; do not open the checkpoint data set
```

Usage notes

The FTP client opens the checkpoint data set for a GET or MGET request when the following conditions are met:
- The data type is EBCDIC
- The file type is SEQ
- The transmission mode is either block or compressed
- The UNIXFILETYPE value is FILE when the local file is a z/OS UNIX file

Guideline: Use RESTGET FALSE to prevent the open of the data set. If the data set is not opened, a failed data transfer in block or compressed mode cannot be restarted.

Related topics

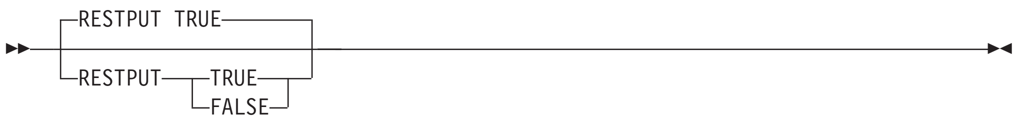
- "CHKPTINT (FTP client and server) statement" on page 762
- "CHKPTPREFIX (FTP client) statement" on page 764
- See REStart subcommand information in z/OS Communications Server: IP User's Guide and Commands.
- "UNIXFILETYPE (FTP client and server) statement" on page 891

RESTPUT (FTP server) statement

Use the RESTPUT statement to specify whether the server supports checkpoint and restart processing when receiving data (put operation).

Server This setting applies when the server is the receiving site.

Syntax



Parameters

TRUE
Specifies that the server supports checkpoint and restart processing when receiving data. This is the default.

FALSE
Specifies that the server does not support checkpoint and restart processing when receiving data. This means that restart markers sent by the client are not supported. When this value is specified, a failed data transfer in block or compressed mode cannot be restarted.

Examples

Use the following code to specify that checkpoint and restart processing should not be supported when the server is receiving data:

```
RESTPUT FALSE
```

Related topic

- "CHKPTINT (FTP client and server) statement" on page 762

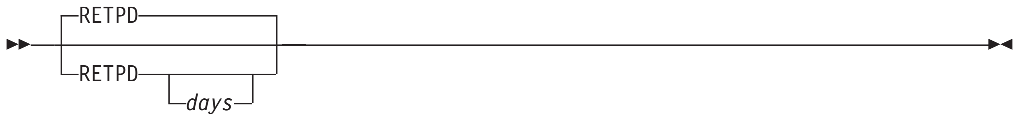
RETPD (FTP client and server) statement

Use the RETPD statement to specify the number of days a newly allocated data set should be retained. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax



Parameters

days

The number of days a newly allocated data set should be retained. The valid range is 0 - 9 999. The default is no retention period assigned to the data set.

If you specify 0 for *days*, newly allocated data sets are assigned a retention period of 0 days. This means that the retention period of the data set expires on the same day that the data set is created.

If you do not specify the RETPD statement or if you specify the RETPD statement with no value, no retention period is assigned to newly allocated data sets.

However, you should understand that the retention period attribute can be obtained from an SMS data class (DATACLASS), an SMS management class (MGMTCLASS), a model data set (DCBDSN), or from the RETPD statement.

You should specify no value for *days* if one of the following situations is true:

- The DATACLASS statement is specified and the retention period from the SMS data class is to be used.
- The MGMTCLASS statement is specified and the retention period from the SMS management class is to be used.
- The DCBDSN statement is specified and the retention period from the model data set is to be used.

If you specify RETPD with a value, this value overrides the retention period settings from any specified model data set (DCBDSN) or SMS data class (DATACLASS) and might override the value of a specified SMS management class (MGMTCLASS).

You should specify no value for days if one of the following situations is true:

- The DATACLASS statement is specified and the retention period from the SMS data class is to be used.
- The MGMTCLASS statement is specified and the retention period from the SMS management class is to be used.
- The DCBDSN statement is specified and the retention period from the model data set is to be used.

If you specify RETPD with no value, and you specified both an SMS data class and a model data set, then the retention period is obtained form the model data set.

If the SMS data class or DCBDSN model data set have a retention period, this retention period can be overridden to a new retention period. The retention period cannot be overridden to have no assigned retention period.

If you specify a management class, then the retention period is obtained from the management class. The value of the management class's retention period can be overridden.

- If a data class is specified, the retention period in the data class can override it.
- If a model data set (DCBDSN) is specified, its retention period overrides both the data class value and the management class value.
- If you specify RETPD with a value, the value you specify overrides any data class setting, model data set value, and any management class setting.

However, regardless of where the retention period value is obtained, when attempting to override the value set in the management class, the actual resulting retention period setting depends on the retention period limit defined in the management class. A management class is defined with a retention limit value as well as a retention period. If you attempt to override the management class's retention period, the override value must be within the retention period limit defined in the management class. Otherwise, the retention period used is the management class's retention limit value.

Examples

- Make the new data set expiration date equal to 30 days:
  ```
  RETPD 30
  ```
- Use a retention period of 0 days:
  ```
  RETPD 0
  ```

Related topics

- "DATACLASS (FTP client and server) statement" on page 770
- "DCBDSN (FTP client and server) statement" on page 776
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- "MGMTCLASS (FTP client and server) statement" on page 818
- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.

SBDATACONN (FTP client and server) statement

This statement defines the conversions between file system code pages and network transfer code pages to be used for data transfer. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server Specifies the single-byte code pages used by the server for data connections.

Client Specifies the single-byte code pages used by the client for data connections.

Syntax

```
►►──SBDATACONN──┬─dsname───────────────────────────────────────────┬───────────►◄
                ├─(file_system_codepage,network_transfer_codepage)─┤
                └─FTP_STANDARD_TABLE───────────────────────────────┘
```

Parameters

dsname
The fully qualified name of an MVS data set or z/OS UNIX file containing the file system to network transfer translate table and the network transfer to file

system translate table generated by the CONVXLAT utility. For more information about translation tables, see Appendix A, "Translation tables," on page 1417.

`file_system_codepage`
The name of a code page that is recognized by iconv. The code page is used for data that is written in the file system.

`network_transfer_codepage`
The name of a code page that is recognized by iconv. The code page is used for data that is transferred on the network.

`FTP_STANDARD_TABLE`
Indicates that the FTP internal tables, which are the same as the tables that are shipped in TCPXLBIN(STANDARD), are to be used.

Examples

```
SBDATACONN (IBM-037,IBM-858)
```

Usage notes

- If you specify SBDATACONN (file_system_codepage,network_transfer_codepage), FTP uses the iconv() application programming interface to translate between the two code pages. The values that you enter on the SBDATACONN statement are used by FTP as parameters to the C++ runtime function iconv(). You can find a valid list of code sets in the z/OS XL C/C++ Programming Guide. See "SBSUB (FTP client and server) statement" on page 843 and "SBSUBCHAR (FTP client and server) statement" on page 844 for more information about using substitution characters to replace unmapped code points during the data transfer.
- The SYSFTSX DD statement, if present, overrides the SBDATACONN statement.
- If neither the SYSFTSX DD statement nor the SBDATACONN statement is present, the search order for a TCPXLBIN data set is followed. See "SBCS translation table hierarchy" on page 1418 for this search order. If no TCPXLBIN data set is found, the same conversion established for the control connection is used for single-byte data transfer.

Related topics

- For the code pages supported by iconv(), see z/OS XL C/C++ Programming Guide.
- "SBSUBCHAR (FTP client and server) statement" on page 844
- "SBSENDEOL statement (FTP client and server) statement"
- "SBSUB (FTP client and server) statement" on page 843

SBSENDEOL statement (FTP client and server) statement

Use the SBSENDEOL statement to tell the FTP client or server what end-of-line (EOL) sequence to use for outbound data when ENcoding is SBCS, Mode is stream, and Type is ASCII. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server Tell the server what EOL sequence to append to each line of text when ENcoding is SBCS, Type is ASCII, and files are sent from the server to the client.

Client Tell the client what EOL sequence to append to each line of text when ENcoding is SBCS, Type is ASCII, and files are sent from the client to the server.

Syntax

```
     ┌─SBSENDEOL CRLF──────────┐
▶▶───┤                         ├──────────────────────────────────────────────────▶◀
     └─SBSENDEOL───┬─CRLF─┬────┘
                   ├─CR───┤
                   ├─LF───┤
                   └─NONE─┘
```

Parameters

CRLF
When translating outbound single-byte data to ASCII, append a carriage return (x'0D') and line feed (x'0A') to each line of text. This is the default and the standard line terminator defined by RFC 959. The z/OS server and client can receive ASCII data in this format only. It is the only setting permitted when using the SRestart subcommand in the client.

CR When translating outbound single-byte data to ASCII, append only a carriage return (x'0D') to each line of text.

LF When translating outbound single-byte data to ASCII, append only a line feed (x'0A') to each line of text.

NONE
When translating outbound single-byte data to ASCII, append no EOL sequence.

Examples

When translating outbound single-byte data to ASCII, to append LF only to each line use the following code:

```
SBSENDEOL  LF
```

To translate files sent from the FTP client to ASCII, without appending an EOL sequence to each line, code the following statements in the client's FTP.DATA. At login, the data type is ASCII and the mode is Stream unless you change the values using subcommands.

```
ENCODING SBCS
SBSENDEOL NONE
```

Restrictions:

- This statement applies only to the end-of-line sequence used on the data connection. The control connection end-of-line sequence is not affected.
- Double-byte, UCS-2, and multi-byte file transfers are not affected by this setting.
- This statement applies only when ENCODING is SBCS, Type is ASCII, and Mode is Stream.

Rule: The SBSENDEOL setting CRLF is the default and the standard EOL sequence defined by RFC 959. It is appropriate for most file transfers. Do not use an alternate SBSENDEOL setting unless you have verified that the recipient FTP can handle the alternate value.

Client Do not code an alternate SBSENDEOL value if your server is a z/OS FTP server. The z/OS FTP server does not support alternate SBSENDEOL values for inbound file transfer.

Server Do not code an alternate SBSENDEOL value if your client is a z/OS FTP client. The z/OS FTP client does not support alternate SBSENDEOL values for inbound file transfer.

Result for FTP client: If you put a file while TYPE is ASCII, MODE is STREAM, ENCODING is SBCS, and SBSENDEOL is not CRLF, the srestart put subcommand is disabled.

Results for FTP server:

- If you code a SBSENDEOL value other than CRLF, the SIZE command is disabled.
- If you transfer a file from the server while TYPE is ASCII, MODE is STREAM, ENCODING is SBCS, and SBSENDEOL is not CRLF, the SIZE command is disabled for the remainder of the session, and the command sequence REST - RETR is disabled for MODE STREAM, TYPE ASCII, ENCODING SBCS file transfers. This precludes stream-mode restart of file transfer to and from the server.
- The REST command in Mode B (Block mode) is not affected by this setting.

Related topics

- "SBDATACONN (FTP client and server) statement" on page 840
- "ENCODING (FTP client and server) statement" on page 790

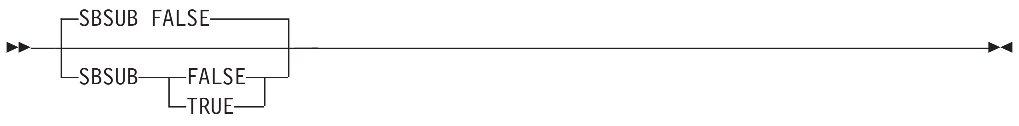
SBSUB (FTP client and server) statement

Use the SBSUB statement in the server and client FTP.DATA to specify whether a substitution is allowed for data bytes that cannot be translated. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server Specifies the whether substitution is allowed on the server's system.

Client Specifies the whether substitution is allowed on the client's system.

Syntax



Parameters

FALSE
Substitution is not allowed for single-byte character translation. This causes a data transfer failure if a character cannot be mapped during the transfer. This is the default value.

TRUE
Substitution is allowed for single-byte character translation. The SBSUBCHAR statement defines the substitution value for untranslatable characters.

Examples

To disable substitution for single-byte character translation, code the following:

```
SBSUB   FALSE
```

Related topics

- “SBDATACONN (FTP client and server) statement” on page 840
- “SBSUBCHAR (FTP client and server) statement”

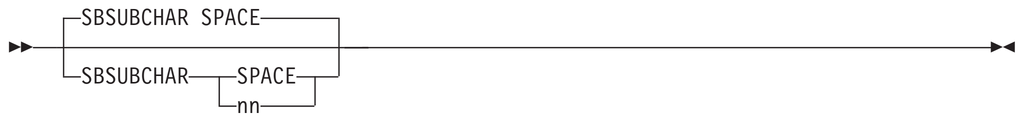
SBSUBCHAR (FTP client and server) statement

Use the SBSUBCHAR statement in the server and client FTP.DATA to specify the substitution character for data transfers using SBCS encodings when SBSUB has a value of TRUE. You can also use the SIte and LOCSIte subcommands to set this keyword.

Server Specifies the substitution character on the server's system.

Client Specifies the substitution character on the client's system.

Syntax



Parameters

SPACE
: Specifies x'40' when target code set is an EBCDIC code set and x'20' when target code set is an ASCII code set. This is the default value.

nn Hexadecimal value that represents a single-byte character. The value of nn can be from 00 to FF.

Examples

To indicate the substitution character to be x'40', ues the following code:

```
SBSUBCHAR  40
```

Related topics

- “SBDATACONN (FTP client and server) statement” on page 840
- “SBSUB (FTP client and server) statement” on page 843

SBTRANS (FTP client) statement

Use the SBTRANS statement to specify the SBCS translation table to be used for the data connection. This table is used for SBCS and DBCS data transfers. FTP uses the translation table in the user_id.dsn_qual.TCPXLBIN data set. If the user*_id.dsn_*qual.TCPXLBIN data set does not exist, FTP uses the *hlq.dsn_qua*l.TCPXLBIN data set.

Syntax

```
►►──SBTRANS──dsn_qual────────────────────────────────────────────────────►◄
```

Parameters

dsn_qual
Specifies the data set qualifier used to name the translation table.

Examples

```
SBTRANS  DATA  ;  use USER33.DATA.TCPXLBIN when ftp
               ;  is used by USER33
```

Usage notes

SBDATACONN and SBTRANS are mutually exclusive statements. If both statements appear in the FTP.DATA file, SBTRANS is ignored.

Related topics

"SBDATACONN (FTP client and server) statement" on page 840

SECONDARY (FTP client and server) statement

Use the SECONDARY statement to specify the number of tracks, blocks, or cylinders (according to SPACETYPE) for secondary allocation.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax

```
   ┌─SECONDARY 1──────────┐
►►─┼──────────────────────┼──────────────────────────────────────────────►◄
   └─SECONDARY──┬────────┬┘
                └─amount─┘
```

Parameters

amount
The number of tracks, blocks, or cylinders. The valid range is 0 - 16 777 215 blocks (the operating system maximum). The default is 1.

- If you specify no value for *amount*, FTP does not specify the number of tracks, blocks, or cylinders for secondary allocation.
- You should specify no value for *amount* if the DATACLASS statement is specified and the space allocation from the SMS data class is to be used. If the SMS data class is to be used for space allocation, both the PRIMARY and SECONDARY values must be omitted, and the value on the SPACETYPE statement is ignored.

  Restriction: If a UNIX file (such as /etc/ftp.data) is being used as the configuration input and no value for the *amount* parameter is specified, then the statement should not have any trailing blanks. Ensure that the line ends after the SECONDARY keyword or that a comment is also specified.

Examples

Set the secondary allocation to two tracks:

```
SECONDARY 2
```

Related topics

- "DATACLASS (FTP client and server) statement" on page 770
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- "PRIMARY (FTP client and server) statement" on page 830
- "SPACETYPE (FTP client and server) statement" on page 873

SECURE_CTRLCONN (FTP client and server) statement

Use the SECURE_CTRLCONN statement to indicate the security level for a control connection. This statement applies only to Kerberos.

Requirement: When using TLS, the control connection must be enciphered and this setting has no effect on the TLS behavior.

Terminology

Integrity protected, data integrity, or data authentication
Indicates that an algorithm is applied to the data being transferred, which modifies that data such that the receiving program can verify the data was not modified or changed during the transfer.

Privacy protected
Indicates that an algorithm is applied to the data being transferred, which encrypts or scrambles the data such that only the receiving program can use a special key to decrypt or unscramble the data to its original format. The original data cannot be seen or interpreted while the data is in transit.

Raw Indicates that data is transmitted without being modified by any encryption or data integrity algorithms.

Encipher or cipher algorithm
Indicates that data being transferred is encrypted, integrity protected, or both. This term does not imply which algorithm is used and does not imply that it is encrypted.

Syntax

```
     ┌─SECURE_CTRLCONN CLEAR──────────┐
►►───┼────────────────────────────────┼─────────────────────────────────────►◄
     └─SECURE_CTRLCONN───┬─CLEAR───┬──┘
                         ├─PRIVATE─┤
                         └─SAFE────┘
```

Parameters

Configuring an FTP server

CLEAR
Specifies that the client decides whether data is transferred raw, integrity protected only, or both integrity and privacy protected.

PRIVATE
Specifies that the server requires data to be transferred using both integrity and privacy protection. Clients attempting to send raw data or data integrity protect only are rejected.

SAFE
Specifies that the server requires data to be transferred using integrity protection only, or using both integrity and privacy protection. Clients attempting to send raw data are rejected.

Configuring an FTP client

CLEAR
Specifies that data can be transferred raw, integrity protected only, or both integrity and privacy protected.

By default, data is transferred raw. However, you can issue the **cprotect private** and **cprotect safe** commands during the FTP session to change the control connection security level. Issuing the **cprotect private** command changes the control connection security level so data is transferred both integrity and privacy protected. Issuing the **cprotect safe** command changes the control connection security level so data is transferred integrity protected only. Then, you can also issue the **cprotect clear** command to reset the control connection security level back, so that data is transferred raw again.

PRIVATE
Specifies that the client data is transferred both integrity and privacy protected.

SAFE
Specifies that the data can be transferred integrity protected only, or both integrity and privacy protected.

By default, data is transferred integrity protected only. However, the client can issue the **cprotect private** during the FTP session to change the control connection security level so data is transferred both integrity and privacy protected. The use can also issue the **cprotect safe** command to reset the control connection security level back, so that data is transferred integrity protected only.

Examples

```
SECURE_CTRLCONN PRIVATE
```

Requirements:

- You must code EXTENSIONS AUTH_GSSAPI for this statement to be used by the FTP server.
- You must code SECURE_MECHANISM GSSAPI for this statement to be used by the FTP client.

Restriction: This statement is ignored when the security mechanism is TLS.

Related topic

- “EXTENSIONS (FTP client and server) statement” on page 792

SECURE_DATACONN (FTP client and server) statement

Use the SECURE_DATACONN statement to indicate the level of security used on data connections, and it applies to both TLS and Kerberos.

See “SECURE_CTRLCONN (FTP client and server) statement” on page 846 for an explanation of terminology for protected, raw, and enciphered data.

Syntax

```
     ┌─SECURE_DATACONN CLEAR──────────┐
►►───┼────────────────────────────────┼──────────────────────────────────────►◄
     └─SECURE_DATACONN──┬─CLEAR───┬───┘
                        ├─NEVER───┤
                        ├─PRIVATE─┤
                        └─SAFE────┘
```

Parameters

Configuring an FTP server

NEVER
: Indicates the server requires data to be transferred raw with no cipher algorithm applied to the data. Clients attempting to use ciphers are rejected.

CLEAR
: Indicates the client decides whether data is transferred raw or enciphered.

  For TLS, the client decides whether data is enciphered or not. If it indicates it should be enciphered, the cipher algorithm is chosen using TLS protocols.

  For Kerberos, the client can specify whether data is transferred raw, integrity protected only, or both integrity and privacy protected.

PRIVATE
: Indicates the server requires data to be transferred enciphered. Clients attempting to send raw data are rejected.

  For TLS, the cipher algorithm is chosen using TLS protocols.

  For Kerberos, the data must be transferred using both integrity and privacy protection. Clients attempting to send data that is only integrity protected are rejected.

SAFE
: For TLS, specifying this option is identical to the PRIVATE specification.

  For Kerberos, the data must be transferred using both integrity and privacy protected. Clients attempting to send data that is only integrity protected are rejected.

Configuring an FTP client

NEVER
: Indicates the client requires data to be transferred raw with no cipher algorithm applied to the data.

CLEAR
: Indicates the data can be transferred raw or enciphered.

  By default, data is transferred raw. However, you can issue the **private** command during the FTP session to change the data connection security level so the data is enciphered. You can also issue the **clear** command to reset the data connection security level back, so that data is transferred raw again.

  For TLS, if the **private** command is issued, the cipher algorithm is chosen using TLS protocols.

For Kerberos, if the **private** command is issued, data is transferred both integrity and privacy protected. In addition to the **private** and **clear** commands, you can issue the **safe** command to change the data connection security level so data is transferred integrity protected only.

PRIVATE
Indicates the client requires data to be transferred enciphered.

For TLS, the cipher algorithm is chosen using TLS protocols.

For Kerberos, the data must be transferred using both integrity and privacy protected.

SAFE
For TLS, specifying this option is identical to the PRIVATE specification.

For Kerberos, the data can be transferred integrity protected only, or both integrity and privacy protected. By default, data is transferred integrity protected only. However, you can issue the **private** command during the FTP session to change the data connection security level so data is transferred both integrity and privacy protected. You can also issue the **safe** command to reset the data connection security level back, so data is transferred integrity protected only.

Examples

```
SECURE_DATACONN NEVER
```

Usage notes

If the FTP server uses the secure port, the server behaves as if the value on this statement is PRIVATE. See "TLSPORT (FTP client and server) statement" on page 880 for information about the secure port.

SECURE_FTP (FTP client and server) statement

Use the SECURE_FTP statement to specify whether use of a security mechanism is optional or required.

Syntax

```
   ┌─SECURE_FTP ALLOWED───────┐
►►─┼──────────────────────────┼──────────────────────────────────────────────►◄
   └─SECURE_FTP──┬─ALLOWED──┬─┘
                 └─REQUIRED─┘
```

Parameters

Configuring an FTP server

REQUIRED
Specifies that all clients log in using a security mechanism.

Rules:

- If the server is enabled for TLS only, clients must log in using TLS.
- If the server is enabled for Kerberos only, clients must log in using Kerberos.
- If the server is enabled for both TLS and Kerberos, clients must log in using either TLS or Kerberos.

ALLOWED

Allows clients to log in using a security mechanism, but it is not required.

Rules:

- If the server is enabled for TLS only, clients must log in using TLS or no securiy mechanism.
- If the server is enabled for Kerberos only, clients must log in using Kerberos or no security mechanism.
- If the server is enabled for both TLS and Kerberos, clients must log in using TLS, Kerberos, or no security mechanism.

Configuring an FTP client

REQUIRED

Specify that a client log in must use a security mechanism. If the server does not support the client's security mechanism, the login fails and the client cannot log in.

Rules:

- If the client's security mechanism is TLS, clients must log in using TLS.
- If the client's security mechanism is Kerberos, clients must log in using Kerberos.

ALLOWED

Allow the client to log in using a security mechanism, but it is not required.

Rules:

- If the client's security mechanism is TLS, clients must log in using TLS. If the server does not support TLS, the server indicates this back to the client. The client then completes the log in, but without using TLS.
- If the client's security mechanism is Kerberos, clients must log in using Kerberos. If the server does not support Kerberos, the server indicates this back to the client. The client then completes the log in, but without using Kerberos.

Examples

```
SECURE_FTP ALLOWED
```

Usage notes

- If the FTP server used the secure port, the server behaves as if the value on this statement is required. See "TLSPORT (FTP client and server) statement" on page 880 for information about the secure port.
- This statement is valid for FTP servers if either EXTENSIONS AUTH_TLS or EXTENSIONS AUTH_GSSAPI is specified.
- This statement is valid for FTP clients if either SECURE_MECHANISM TLS or SECURE_MECHANISM GSSAPI is specified.

Related topics

- "SECURE_MECHANISM (FTP client) statement" on page 854
- "EXTENSIONS (FTP client and server) statement" on page 792
- See z/OS Communications Server: IP Configuration Guide for more information about customizing TLS and Kerberos and SSL/TLS.

SECURE_HOSTNAME (FTP client) statement

Use the SECURE_HOSTNAME statement to specify whether the client verifies the host name in the server's certificate.

The statement is ignored for sessions that are not protected by the TLS security mechanism.

Syntax

```
   ┌─SECURE_HOSTNAME OPTIONAL──────────┐
►►─┴─SECURE_HOSTNAME──┬─OPTIONAL─┬─────┴──────────────────────────────────────►◄
                      └─REQUIRED─┘
```

Parameters

REQUIRED
Specifies that the host name that the client is connecting to is verified against the server's certificate. Either the common name or the subject alternate name contained in the server's X.509 certificate is used to validate the host name. If the verification fails, the connection is terminated.

OPTIONAL
Specifies that the host name is not validated. This is the default.

SECUREIMPLICITZOS (FTP client and server) statement

Use the SECUREIMPLICITZOS statement to specify when FTP should negotiate or expect the security handshake for TLSPORT implicitly secured connections.

Rules:

- To enable a z/OS FTP client to log into the z/OS FTP server using the protected port, specify the same SECUREIMPLICITZOS statement value and TLSPORT value for the client and server.
- When using the implicit connection (FTP client is connecting to the port specified by the TLSPORT statement), some FTP servers expect to negotiate the security of the session immediately after the connection is issued. If you are initiating a secure session with such a server, code SECUREIMPLICITZOS FALSE in the client's FTP.DATA file.
- Many non-z/OS FTP clients negotiate the security immediately after the connect and before the initial 220 reply is received from the server. To enable these clients to log into the z/OS FTP server's protected port, code SECUREIMPLICITZOS FALSE in the server's FTP.DATA file.

Server The first reply that the FTP server sends to a client uses reply code 220. The reply is sometimes referred to as the good morning reply. The SECUREIMPLICITZOS statement specifies whether the server expects the TLS handshake to occur before or after it sends the initial reply 220.

Client The SECUREIMPLICITZOS statement specifies when the client initiates the TLS handshake for connections to the TLSPORT (protected port). You can change this setting using the locsite subcommand.

Syntax

```
     ┌─SECUREIMPLICITZOS TRUE──────────┐
►►───┤                                 ├──────────────────────────────────────────►◄
     └─SECUREIMPLICITZOS──┬─TRUE──┬────┘
                          └─FALSE─┘
```

Parameters

TRUE

This is the default.

Server Specifies that the FTP server expects the security handshake to occur after it sends the reply 220.

Client Specifies that the FTP client initiates the security handshake after the 220 (good morning) reply is received from the server.

FALSE

Server Specifies that the FTP server expects the security handshake before it sends the reply 220.

Client Specifies that the FTP client negotiates the security handshake immediately after the connection and before the initial 220 reply is received from the server.

Examples

To initiate an implicitly secured session between a z/OS FTP client and a z/OS FTP server, code the following statements in the FTP client and server FTP.DATA file:

```
SECUREIMPLICITZOS  TRUE
```

You could also code the following statement in both the FTP client and server FTP.DATA file:

```
SECUREIMPLICITZOS  FALSE
```

To initiate an implicitly secured session between a non-z/OS FTP client and a z/OS FTP server, code the following statement in the FTP server FTP.DATA file:

```
SECUREIMPLICITZOS FALSE
```

Related topic

- "TLSPORT (FTP client and server) statement" on page 880

SECURE_LOGIN (FTP server) statement

Use the SECURE_LOGIN statement to indicate whether the FTP server requires client authentication.

The SECURE_LOGIN statement setting applies to TLS and Kerberos. Note that the term certificate is actually TLS terminology. In Kerberos, the equivalent of a certificate is a ticket, which contains credentials.

Rules:

- This statement is valid only when you have coded EXTENSIONS TLS or EXTENSIONS AUTH in the FTP.DATA file of the server.
- If you code VERIFYUSER TRUE in FTP.DATA, the server verifies the user's access to the FTP server port profile in the SERVAUTH class regardless of the SECURE_LOGIN value.

Syntax

```
      ┌─SECURE_LOGIN NO_CLIENT_AUTH──────┐
►►────┤                                  ├─────────────────────────────────►◄
      └─SECURE_LOGIN──┬─VERIFY_USER────┬─┘
                      ├─NO_CLIENT_AUTH─┤
                      └─REQUIRED───────┘
```

Parameters

VERIFY_USER

Indicates that in addition to client certificate authentication, the user's ID is further verified.

For Kerberos, the user ID in the client's ticket is verified to match the login user ID.

```
EZB.FTP.MVS164.FTPD1.PORT21
```

For TLS:

- The server verifies that the certificate has been registered with your SAF-compliant security product, such as RACF, and has an associated user ID matching the login user ID.
- If the SERVAUTH RACF (or another security product) class is active and a RACF resource has been defined for the port, the connection is allowed only if the user ID associated with the client certificate has READ access to the RACF resource.

  The resource name would be:

  ```
  EZB.FTP.<systemname>.<ftpdaemonname>.PORTxxxx
  ```

  where xxxx is replaced by the port number for the FTP daemon. For example, if the procedure FTPD is used to start the daemon on system MVS164 and the daemon uses the default FTP port 21, then the resource name is:

  ```
  EZB.FTP.MVS164.FTPD1.PORT21
  ```

Tip: For sessions that are not secured with TLS, you can use the same resource profile to control which users can log into the FTP server when you code VERIFYUSER TRUE in the server's FTP.DATA file. However, if you do code VERIFYUSER TRUE in FTP.DATA, the server verifies the user's access to the resource profile regardless of the SECURE_LOGIN value.

REQUIRED

Indicates that the server should authenticate client certificates.

This does not affect Kerberos behavior; Kerberos always processes the client's ticket.

For TLS, client certificate authentication occurs during the SSL handshake. To pass authentication, the Certificate Authority (CA) that signed the client certificate must be considered trusted by the server. This means a certificate for the CA that issued the client certificate is listed as trusted in the server's key ring.

NO_CLIENT_AUTH
Specifies that the server should not request the client certificate for TLS.

This parameter has no effect for Kerberos.

Examples

```
SECURE_LOGIN REQUIRED
```

Related topics

- “EXTENSIONS (FTP client and server) statement” on page 792
- “SECURE_PASSWORD (FTP server) statement” on page 855
- “SECURE_PASSWORD_KERBEROS (FTP server) statement” on page 856
- “VERIFYUSER (FTP server) statement” on page 893

SECURE_MECHANISM (FTP client) statement

Use the SECURE_MECHANISM statement to specify whether the FTP client should use a security mechanism when a session is established. The parameter on the statement indicates which security mechanism to use.

Syntax

```
►►──SECURE_MECHANISM──┬─TLS────┬──────────────────────────────────────►◄
                      └─GSSAPI─┘
```

Parameters

TLS
Specifies that TLS is the security mechanism that is used by the client when it establishes a session.

GSSAPI
Specifies that GSSAPI is the security mechanism that is used by the client when it establishes a session.

Examples

To specify that TLS protocols should be use for the session, use the following code:

```
SECURE_MECHANISM  TLS
```

Usage notes

- Security mechanism GSSAPI is supported for IPv4 connections only.
- The SECURE_MECHANISM statement can be overridden by the -a or the -r start parameter on the FTP command.

Related topics

- “SECURE_FTP (FTP client and server) statement” on page 849
- See the FTP command and the FTP environment information in z/OS Communications Server: IP User's Guide and Commands.

- See z/OS Communications Server: IP Configuration Guide for more information about customizing TLS and Kerberos and SSL/TLS.

SECURE_PASSWORD (FTP server) statement

Use the SECURE_PASSWORD statement to specify whether a password is required by the FTP server for an TLS protected session. The statement is ignored for sessions that are not protected by the TLS security mechanism.

Syntax

```
   ┌─SECURE_PASSWORD REQUIRED────────┐
►►─┼─────────────────────────────────┼──────────────────────────────────────►◄
   └─SECURE_PASSWORD──┬─REQUIRED─┬───┘
                      └─OPTIONAL─┘
```

Parameters

REQUIRED
: Specifies that a password is required to log in a user whose session is protected by the TLS security mechanism.

OPTIONAL
: Specifies that the password is not required if the client provides a certificate that can be used to authenticate the user. See the Usage notes in this topic for more information.

  If the client certificate is used to authenticate the user and the authentication fails, the login attempt fails.

Rule: The handshake that occurs when the TLS protected session is established must include the transfer of the client certificate to the server. If you code SECURE_PASSWORD OPTIONAL, you must code SECURE_LOGIN VERIFY_USER or SECURE_LOGIN REQUIRED to require the client certificate.

Result: If you code SECURE_PASSWORD OPTIONAL and SECURE_LOGIN NO_CLIENT_AUTH in the FTP.DATA file, the message EZYFS16I is logged to inform you that the combination is not allowed. The value set by the SECURE_PASSWORD statement is changed to REQUIRED.

Examples

To require the user to enter a password on an TLS protected session only when the USER name does not match the name associated with the certificate, code the following statements:

```
SECURE_LOGIN      REQUIRED
SECURE_PASSWORD   OPTIONAL
```

Usage notes

The certificate that is received from the client must be registered in the security product and must be associated with the user ID that is passed on the USER command to the FTP server. You can use RACDCERT ADD command to register and associate the certificate.

When the certificate is registered in the security product and is associated with the user ID that is passed in on the USER command, the SECURE_PASSWORD statement value determines the action taken during the login procedure.

Table 49 shows the statement value options.

Table 49. SECURE_PASSWORD statement value options

SECURE_PASSWORD	**SECURE_LOGIN**	**Action**
REQUIRED	VERIFY_USER or REQUIRED	Prompt for a password.
OPTIONAL	VERIFY_USER or REQUIRED	Authenticate with the certificate (do not prompt for password if the authenticate fails).

When either the certificate is not registered in the security product or is not associated with the user ID that is passed in on the USER command, the SECURE_LOGIN statement value determines the action during the login procedure.

Table 50 shows the statement value options.

Table 50. SECURE_LOGIN statement value options

SECURE_PASSWORD	**SECURE_LOGIN**	**Action**
REQUIRED or OPTIONAL	VERIFY_USER	Fail the login.
REQUIRED or OPTIONAL	REQUIRED	Prompt for a password.

Related topics

- “EXTENSIONS (FTP client and server) statement” on page 792
- “SECURE_LOGIN (FTP server) statement” on page 852
- See z/OS Communications Server: IP Configuration Guide for more information about customizing TLS and Kerberos and SSL/TLS.

SECURE_PASSWORD_KERBEROS (FTP server) statement

Use the SECURE_PASSWORD_KERBEROS statement to specify whether a password is required by the FTP server for a Kerberos-protected session. The statement is ignored for sessions that are not protected by the Kerberos security mechanism.

Rule: This statement is enabled only when EXTENSIONS AUTH_GSSAPI is coded in the server's FTP.DATA file.

When the user ID passed on the USER command matches the user ID that the SAF-compliant security product maps to the user ID that the Kerberos principal received from the client, the SECURE_PASSWORD_KERBEROS statement value determines whether the server prompts the client for the password during the login procedure.

Syntax

```
      ┌─SECURE_PASSWORD_KERBEROS REQUIRED──────┐
►►────┤                                        ├──────────────────────────────────►◄
      └─SECURE_PASSWORD_KERBEROS──┬─REQUIRED─┬─┘
                                  └─OPTIONAL─┘
```

Parameters

REQUIRED
Specifies that a password is required to log in a user whose session is protected by the Kerberos security mechanism.

This is the default.

OPTIONAL
Specifies that the password is not required if the user ID passed on the USER command matches the user ID that the SAF-compliant security product mapped to the user ID that the Kerberos principal received from the client.

Examples

To require the user to enter a password on a Kerberos-protected session only when the user ID passed on the USER command does not match the user ID that the SAF-compliant security product mapped to the user ID that the Kerberos principal received from the client, code the following statement:

```
SECURE_PASSWORD_KERBEROS   OPTIONAL
```

Usage notes

Table 51 shows how the SECURE_PASSWORD_KERBEROS statement affects user authentication when the user ID to which the Kerberos principal is mapped matches the user ID that is passed on the USER command.

Table 51. User identity in the Kerberos ticket matches user ID on USER command

SECURE_PASSWORD_KERBEROS	SECURE_LOGIN	Action
REQUIRED	One of the following: • VERIFY_USER • REQUIRED • NO_CLIENT_AUTH	Prompt for a password.
OPTIONAL	One of the following: • VERIFY_USER • REQUIRED • NO_CLIENT_AUTH	Authenticate with the Kerberos ticket (if the Kerberos authentication fails, fail the login, do not prompt for password).

When the user ID to which the Kerberos principal is mapped does not match the user ID that is passed on the USER command, the SECURE_LOGIN statement value determines the action that is necessary during the authentication procedure.

Table 52 shows how the SECURE_LOGIN statement affects user authentication when the user ID to which the Kerberos principal is mapped does not match the user ID that is passed on the USER command.

Table 52. User identity in the Kerberos ticket does not match user ID on USER command

SECURE_PASSWORD_KERBEROS	**SECURE_LOGIN**	**Action**
REQUIRED or OPTIONAL	VERIFY_USER	Fail the login.
REQUIRED or OPTIONAL	REQUIRED or NO_CLIENT_AUTH	Prompt for a password.

Related topics

- "EXTENSIONS (FTP client and server) statement" on page 792
- "SECURE_LOGIN (FTP server) statement" on page 852
- See z/OS Communications Server: IP Configuration Guide for more information about customizing TLS and Kerberos and SSL/TLS.

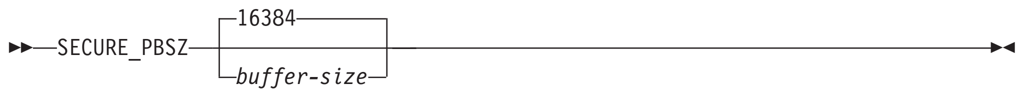
SECURE_PBSZ (FTP client and server) statement

Specifies the maximum size of the encoded data blocks sent during file transfer.

Server Specifies the maximum protection buffer size the server accepts.

Client Specifies the protection buffer size the client uses to negotiate with the server.

Syntax



Parameters

buffer_size
The valid range is between 512 - 32 768. The default value is 16 384.

Usage notes

- The client initially issues the PBSZ command specifying buffer_size. If the PBSZ command is rejected, the client reissues the PBSZ command with a smaller value until it is accepted by the server.
- If the server receives a protection buffer size (PBSZ) larger than the value configured in the server's FTP.DATA configuration file, the server defaults to its configured value.

- The setting applies only to the Kerberos protocol.

Related topics

- "EXTENSIONS (FTP client and server) statement" on page 792
- "SECURE_MECHANISM (FTP client) statement" on page 854

SEQNUMSUPPORT (FTP client) statement

Use the SEQNUMSUPPORT statement to ignore sequence numbers in files designated by the ddname INPUT.

Syntax

```
     ┌─SEQNUMSUPPORT FALSE─────────┐
►►───┼─────────────────────────────┼────────────────────────────────────────►◄
     └─SEQNUMSUPPORT ──┬─TRUE──┬───┘
                       └─FALSE─┘
```

Parameters

TRUE
: When reading FTP subcommands to be processed from the ddname INPUT, FTP removes any sequence numbers before processing the command.

FALSE
: When reading FTP subcommands, any sequence numbers in the input designated by the ddname INPUT are considered to be part of the input. This is the default.

Examples

The following example shows how data sets with different sequence number schemes can be concatenated if SEQNUMSUPPORT TRUE is coded in the FTP.DATA file:

```
Suppose dataset:  FTP.SUBCMDS(LOGIN) contains no sequence numbers

  mvs056.tcp.raleigh.ibm.com
  user1
  us3rpswd

Suppose dataset:  FTP.SUBCMDS(FTPINFO) contains TRAILING sequence numbers

  ; This comment prevents 00000100 and subsequent sequence numbers         00000100
  ; from being interpreted as an ftp subcommand.                           00000110
  locstat                                                                  00000120
  stat                                                                     00000130
  pwd                                                                      00000140

  Suppose dataset:  FTP.SUBCMDS(FTPCMDS1) contains no sequence numbers

  ; This comment indicates no sequence numbers present
  get remote.file.name local.name.

  Suppose dataset:  FTP.SUBCMDS(FTPCMDS2) contains LEADING sequence numbers

  00000100; The file indicates leading sequence numbers present
  00000110put local.file +
  00000120 remote.file
```

To specify the datasets listed previously as input to the FTP client, the following sample JCL is used:

```
//FTP  EXEC PGM=FTP
 //SYSPRINT DD SYSOUT=*
 //SYSFTPD DD DSN=SYS1.TCPPARMS(FTPCDATA),DISP=SHR
 //* Insure that SEQNUMSUPPORT TRUE is coded
 //* in the above clients FTP.DATA file
 //INPUT DD DSN=FTP.SUBCMDS(LOGIN),DISP=SHR
 //      DD DSN=FTP.SUBCMDS(FTPINFO),DISP=SHR
 //      DD DSN=FTP.SUBCMDS(FTPCMDS1),DISP=SHR
 //      DD DSN=FTP.SUBCMDS(FTPCMDS2),DISP=SHR
```

Results:

- When SEQNUMSUPPORT TRUE is coded in the FTP.DATA file and the FTP client reads the first record of the file specified by the ddname INPUT, the record determines the type of sequence numbers that are to be processed.
- If the last eight columns are numeric and contain trailing sequence numbers, this data is replaced with blanks before running this and subsequent records. Otherwise, if the first eight columns are numeric and contain leading sequence numbers, the data that begins in column 9 is shifted to column 1 before the record is processed.
- If FTP detects no sequence numbers, the data is not modified.
- Each time a semicolon (;) is detected in the first data column, FTP determines the sequencing mode to use to process sequence numbers that follow statement.

Requirements:

- If you concatenate files with the INPUT DD statement, the first statement in each concatenated file must have a semicolon (;) in column 1; the semicolon to enables the FTP client to correctly determine the sequence numbering scheme that is being used.
- If no semicolon (;) is present in the concatenated files, sequence number processing does not change.

SMF (FTP server) statement

Use the SMF statement to specify SMF recording options.

Syntax

```
►►──SMF──┬─STD─────┬──────────────────────────────────────────────────►◄
         ├─number──┤
         └─TYPE119─┘
```

Parameters

STD

Indicates that all FTP server SMF records of type 118 are issued with the following subtypes:

- APPEND - 70
- DELETE - 71
- LOGIN FAILURE - 72
- RENAME - 73
- RETRIEVE - 74

- STORE - 75
- STORE UNIQUE - 75

number
The SMF record subtype to be used for all FTP server records unless otherwise specified for a particular record subtype. The valid range is 1 - 255. There is no default value.

Restriction: This field applies to type 118 records only.

TYPE119
Indicates that all FTP server SMF records of type 119 are issued. Type 119 records have the following subtypes:
- APPEND - 70
- DELETE - 70
- DAEMON CONFIGURATION -71
- LOGIN FAILURE - 72
- RENAME - 70
- RETRIEVE - 70
- STORE - 70
- STORE UNIQUE - 70

Examples

To have all 118 FTP server records created with standard subtypes:

```
SMF STD
```

To have all type 119 FTP server records created:

```
SMF TYPE119
```

To have all type FTP server records of both types created with standard subtypes for type 118 records:

```
SMF STD
SMF TYPE119
```

To log all FTP records of type 119, as well as type 118 APPEND records:

```
SMF TYPE119
SMFAPPE 99
```

To log all FTP records of type 118 with standard subtypes, as well as type 119 DELETE and RENAME records:

```
SMF STD
SMFDEL  TYPE119
SMFREN  TYPE119
```

Usage notes

- SMF statements for each record type (118 and 119) function independently of each other.
- If the SMF statement is omitted, SMF recording occurs for only the events with a statement coded. For example, if SMF is omitted but an SMFAPPE statement is coded, only the APPEND command has SMF recording.
- If the SMF statement is coded with a value of STD, all other SMF-related statements using type 118 records with a value coded (even if it is STD) are

flagged with warning message EZYFT58 and their specifications are ignored. SMF STD means standard type 118 values and no other type 118 values are allowed.

For example, if SMF STD is specified, then specifying SMFAPPE STD is flagged with message `EZYFT58` and is ignored.

- If none of the SMF subtype statements are coded in the FTP.DATA data set, then no SMF records are written by the FTP server.
- Records of type 118 and type 119 can both be requested; however, do not do this due to performance implications of writing both record types. Use type 119 records instead of type 118 records, as type 119 records generally use more standard formatting and provide more information.

Related topics

- "SMFAPPE (FTP server) statement"
- "SMFDCFG (FTP server) statement" on page 863
- "SMFDEL (FTP server) statement" on page 864
- "SMFEXIT (FTP server) statement" on page 865
- "SMFJES (FTP server) statement" on page 866
- "SMFLOGN (FTP server) statement" on page 867
- "SMFREN (FTP server) statement" on page 868
- "SMFRETR (FTP server) statement" on page 869
- "SMFSQL (FTP server) statement" on page 870
- "SMFSTOR (FTP server) statement" on page 871

SMFAPPE (FTP server) statement

Use the SMFAPPE statement to specify the SMF record subtype to be used for the APPE (APPEND) command.

Syntax

```
►►──SMFAPPE──┬─STD─────┬──────────────────────────────────────────────►◄
             ├─number──┤
             └─TYPE119─┘
```

Parameters

STD
: Indicates that type 118 SMF APPEND records are issued with the standard subtype of 70.

number
: Indicates that type 118 SMF APPEND records are issued with the given record subtype. The valid range is 1 - 255.

TYPE119
: Indicates that type 119 SMF APPEND records are issued (subtype 70).

Examples

Set the type 118 SMF record subtype for APPEND to 70:

```
SMFAPPE 70
```

To issue type 119 SMF APPEND records:

```
SMFAPPE TYPE119
```

Usage notes

- SMFAPPE statements for each record type (118 and 119) function independently of each other.
- If you do not specify the SMFAPPE statement for a particular record type (118 or 119), SMF Append records of that type are still issued if the corresponding SMF statement for that record type is present.

Related topics

- “SMF (FTP server) statement” on page 860
- “SMFDCFG (FTP server) statement”
- “SMFDEL (FTP server) statement” on page 864
- “SMFEXIT (FTP server) statement” on page 865
- “SMFJES (FTP server) statement” on page 866
- “SMFLOGN (FTP server) statement” on page 867
- “SMFREN (FTP server) statement” on page 868
- “SMFRETR (FTP server) statement” on page 869
- “SMFSQL (FTP server) statement” on page 870
- “SMFSTOR (FTP server) statement” on page 871

SMFDCFG (FTP server) statement

Use the SMFDCFG statement to specify that a type 119 SMF record of subtype 71 is collected for the FTP daemon configuration information when the FTP daemon starts.

Syntax

```
►►──SMFDCFG──────────────────────────────────────────────────────────►◄
```

Parameters

The SMFDCFG statement has no parameters. If you use the SMFDCGF statement with a parameter, the parameter is ignored.

Examples

To record FTP daemon configuration information when the FTP daemon starts, use the following statement:

```
SMFDCFG
```

Usage notes

- If you do not specify the SMFDCFG statement, the SMF record for the FTP daemon configuration information is issued only when the SMF TYPE119 statement is present. If neither the SMF nor the SMFDCFG statement is specified, no SMF records are collected for the FTP daemon configuration information.
- Only type 119 SMF records are available for FTP daemon configuration information. No corresponding type 118 SMF records are available.

Related topics

- "SMF (FTP server) statement" on page 860
- "SMFAPPE (FTP server) statement" on page 862
- "SMFDEL (FTP server) statement"
- "SMFEXIT (FTP server) statement" on page 865
- "SMFJES (FTP server) statement" on page 866
- "SMFLOGN (FTP server) statement" on page 867
- "SMFREN (FTP server) statement" on page 868
- "SMFRETR (FTP server) statement" on page 869
- "SMFSQL (FTP server) statement" on page 870
- "SMFSTOR (FTP server) statement" on page 871

SMFDEL (FTP server) statement

Use the SMFDEL statement to specify SMF recording options for the DELE (DELETE) command.

Syntax

```
►►──SMFDEL──┬─STD─────┬──────────────────────────────────────────────►◄
            ├─number──┤
            └─TYPE119─┘
```

Parameters

STD
Indicates that type 118 SMF DELETE records are issued with the standard subtype of 71.

number
Indicates that type 118 SMF DELETE records are issued with the given record subtype. The valid range is 1 - 255.

TYPE119
Indicates that type 119 SMF DELETE records are issued (subtype 70).

Examples

Set the type 118 SMF record subtype for DELETE to 71:

```
SMFDEL 71
```

To issue type 119 SMF DELETE records:

```
SMFDEL TYPE119
```

Usage notes

- SMFDEL statements for each record type (118 and 119) function independently of each other. To collect both types, you must specify both SMFDEL STD and SMFDEL TYPE119.
- If you do not specify the SMFDEL statement, SMF records for the DELETE command are still issued if the SMF statement is present. (Type 118 DELETE records have the subtype specified with the SMF statement; type 119 DELETE records are always subtype 70.) If neither the SMF or SMFDEL statement is specified, no SMF records are collected for the DELETE command.

Related topics

- “SMF (FTP server) statement” on page 860
- “SMFAPPE (FTP server) statement” on page 862
- “SMFDCFG (FTP server) statement” on page 863
- “SMFEXIT (FTP server) statement”
- “SMFJES (FTP server) statement” on page 866
- “SMFLOGN (FTP server) statement” on page 867
- “SMFREN (FTP server) statement” on page 868
- “SMFRETR (FTP server) statement” on page 869
- “SMFSQL (FTP server) statement” on page 870
- “SMFSTOR (FTP server) statement” on page 871

SMFEXIT (FTP server) statement

Use the SMFEXIT statement to specify that the user exit routine FTPSMFEX is called before writing the Type 118 SMF record to SMF data sets.

Syntax

```
►►──SMFEXIT──────────────────────────────────────────────────────────►◄
```

Parameters

This statement has no parameters.

Examples

To specify that the user exit FTPSMFEX is called before writing the Type 118 SMF record to SMF data sets, use the following code:

```
SMFEXIT
```

Usage notes

The FTP SMF user exit has been discontinued for type 119 FTP SMF records. The user exit routine FTPSMFEX is only to be called for any type 118 records that are written; no FTP-specific exit is called for type 119 records. In order to obtain the same functionality with type 119 records, the system-wide SMF user exits should now be used (IEFU83, IEFU84, and IEFU85). See z/OS MVS System Management Facilities (SMF) for more information.

Related topics

- “The FTP server SMF user exit” on page 710
- “FTP server user exits” on page 700
- “SMF (FTP server) statement” on page 860
- “SMFAPPE (FTP server) statement” on page 862
- “SMFDCFG (FTP server) statement” on page 863
- “SMFDEL (FTP server) statement” on page 864
- “SMFJES (FTP server) statement” on page 866
- “SMFLOGN (FTP server) statement” on page 867

- “SMFREN (FTP server) statement” on page 868
- “SMFRETR (FTP server) statement” on page 869
- “SMFSQL (FTP server) statement” on page 870
- “SMFSTOR (FTP server) statement” on page 871

SMFJES (FTP server) statement

Use the SMFJES statement to specify that SMF records are collected when FILETYPE is JES (remote job submission).

If SMFJES is not specified, no SMF records are issued when FILETYPE is JES.

Syntax

```
►►──SMFJES──┬─────────┬──────────────────────────────────────────►◄
            └─TYPE119─┘
```

Parameters

TYPE119
: Issue records for filetype JES for all type 119 SMF records. If no parameters are given, records for filetype JES are issued for all type 118 SMF records.

Examples

To record SMF type 118 records for STOR when FILETYPE=JES, use the following code:

```
SMFSTOR STD
SMFJES
```

To record SMF type 119 records for STOR when FILETYPE=JES, use the following code:

```
SMFSTOR TYPE119
SMFJES TYPE119
```

Usage notes

SMFJES statements for each record type (118 and 119) function independently of each other.

Related topics

- “FILETYPE (FTP client and server) statement” on page 796
- “JESINTERFACELEVEL (FTP server) statement” on page 804
- “SMF (FTP server) statement” on page 860
- “SMFAPPE (FTP server) statement” on page 862
- “SMFDCFG (FTP server) statement” on page 863
- “SMFDEL (FTP server) statement” on page 864
- “SMFEXIT (FTP server) statement” on page 865
- “SMFLOGN (FTP server) statement” on page 867
- “SMFREN (FTP server) statement” on page 868
- “SMFRETR (FTP server) statement” on page 869

- “SMFSQL (FTP server) statement” on page 870
- “SMFSTOR (FTP server) statement” on page 871

SMFLOGN (FTP server) statement

Use the SMFLOGN statement to specify the SMF recording options when recording logon failures.

Syntax

```
►►──SMFLOGN──┬─STD─────┬─────────────────────────────────────────►◄
             ├─number──┤
             └─TYPE119─┘
```

Parameters

STD
: Indicates that type 118 SMF logon failure records are issued with the standard subtype of 72.

number
: Indicates that type 118 SMF logon failure records are issued with the given record subtype. The valid range is 1 - 255.

TYPE119
: Indicates that type 119 SMF logon failure records are issued (subtype 72).

Examples

Set the type 118 SMF record subtype for logon failures to 72:

```
SMFLOGN  72
```

To issue type 119 SMF LOGON records:

```
SMFLOGN TYPE119
```

Usage notes

- There is no default value; however, if the SMF statement is coded for type 118 records, the value specified for the SMF statement is used as the default.
- SMFLOGN statements for each record type (118 and 119) function independently of each other.
- If you do not specify the SMFLOGN statement, SMF records for logon failures are still issued if the SMF statement is present (type 118 logon failure records have the subtype specified with the SMF statement; type 119 logon failure records are always subtype 72). If neither the SMF or SMFLOGN statement is specified, no SMF records are collected for logon failures.

Related topics

- “FTP server user exits” on page 700
- “SMF (FTP server) statement” on page 860
- “SMFAPPE (FTP server) statement” on page 862
- “SMFDCFG (FTP server) statement” on page 863
- “SMFDEL (FTP server) statement” on page 864
- “SMFEXIT (FTP server) statement” on page 865

- “SMFJES (FTP server) statement” on page 866
- “SMFREN (FTP server) statement”
- “SMFRETR (FTP server) statement” on page 869
- “SMFSQL (FTP server) statement” on page 870
- “SMFSTOR (FTP server) statement” on page 871

SMFREN (FTP server) statement

Use the SMFREN statement to specify SMF recording options for the RNFR/RNTO (RENAME) command.

Syntax

```
►►──SMFREN──┬─STD─────┬──────────────────────────────────────────────►◄
            ├─number──┤
            └─TYPE119─┘
```

Parameters

STD
: Indicates that type 118 SMF RENAME records are issued with the standard subtype of 73.

number
: Indicates that type 118 SMF RENAME records are issued with the given record subtype. The valid range is 1 - 255.

TYPE119
: Indicates that type 119 SMF RENAME records are issued (subtype 70).

Examples

Set the type 118 SMF record subtype for RENAME to 73:

```
SMFREN 73
```

To issue type 119 SMF RENAME records:

```
SMFREN TYPE119
```

Usage notes

- There is no default value; however, if the SMF statement is coded for type 118 records, the value specified for the SMF statement is used as the default.
- SMFREN statements for each record type (118 and 119) function independently of each other.
- If you do not specify the SMFREN statement, SMF records for the RENAME command is still issued if the SMF statement is present (type 118 RENAME records have the subtype specified with the SMF statement; type 119 RENAME records are always subtype 70). If neither the SMF or SMFREN statement is specified, no SMF records are collected for the RENAME command.

Related topics

- “FTP server user exits” on page 700
- “SMF (FTP server) statement” on page 860
- “SMFAPPE (FTP server) statement” on page 862

- “SMFDCFG (FTP server) statement” on page 863
- “SMFDEL (FTP server) statement” on page 864
- “SMFEXIT (FTP server) statement” on page 865
- “SMFJES (FTP server) statement” on page 866
- “SMFLOGN (FTP server) statement” on page 867
- “SMFRETR (FTP server) statement”
- “SMFSQL (FTP server) statement” on page 870
- “SMFSTOR (FTP server) statement” on page 871

SMFRETR (FTP server) statement

Use the SMFRETR statement to specify SMF recording options for the RETR (RETRIEVE) command.

Syntax

```
►►──SMFRETR──┬─STD─────┬──────────────────────────────────────────────►◄
             ├─number──┤
             └─TYPE119─┘
```

Parameters

STD
Indicates that type 118 SMF RETRIEVE records are issued with the standard subtype of 74.

number
Indicates that type 118 SMF RETRIEVE records are issued with the given record subtype. The valid range is 1 - 255.

TYPE119
Indicates that type 119 SMF RETRIEVE records are issued (subtype 70).

Examples

Set the type 118 SMF record subtype for RETRIEVE to 74:

```
SMFRETR 74
```

To issue type 119 SMF RETRIEVE records:

```
SMFRETR TYPE119
```

Usage notes

- There is no default value; however, if the SMF statement is coded for type 118 records, the value specified for the SMF statement is used as the default.
- SMFRETR statements for each record type (118 and 119) function independently of each other.
- If you do not specify the SMFRETR statement, SMF records for the RETRIEVE command are still issued if the SMF statement is present. (Type 118 RETRIEVE records have the subtype specified with the SMF statement; type 119 RETRIEVE records are always subtype 70.) If neither the SMF or SMFRETR statement is specified, no SMF records are collected for the RETRIEVE command.

Related topics

- “FTP server user exits” on page 700
- “SMF (FTP server) statement” on page 860
- “SMFAPPE (FTP server) statement” on page 862
- “SMFDCFG (FTP server) statement” on page 863
- “SMFDEL (FTP server) statement” on page 864
- “SMFEXIT (FTP server) statement” on page 865
- “SMFJES (FTP server) statement” on page 866
- “SMFLOGN (FTP server) statement” on page 867
- “SMFREN (FTP server) statement” on page 868
- “SMFSQL (FTP server) statement”
- “SMFSTOR (FTP server) statement” on page 871

SMFSQL (FTP server) statement

Use the SMFSQL statement to specify that SMF records are collected when FILETYPE is SQL (SQL query function).

If SMFSQL is not specified, no SMF records are issued when FILETYPE is SQL.

Syntax

```
►►──SMFSQL──┬─────────┬──────────────────────────────────────────────►◄
            └─TYPE119─┘
```

Parameters

TYPE119
: Issue records for filetype SQL for all type 119 SMF records. If no parameters are given, records for filetype SQL are issued for all type 118 SMF records.

Examples

To record SMF type 118 records for RETR when FILETYPE=SQL, use the following code:

```
SMFRETR STD
SMFSQL
```

To record SMF type 119 records for RETR when FILETYPE=SQL, use the following code:

```
SMFRETR TYPE119
SMFSQL TYPE119
```

Usage notes

SMFSQL statements for each record type (118 and 119) function independently of each other.

Related topics

- “DB2 (FTP client and server) statement” on page 774
- “DB2PLAN (FTP cilent and server) statement” on page 775

- “FTP server user exits” on page 700
- “SMF (FTP server) statement” on page 860
- “SMFAPPE (FTP server) statement” on page 862
- “SMFDCFG (FTP server) statement” on page 863
- “SMFDEL (FTP server) statement” on page 864
- “SMFEXIT (FTP server) statement” on page 865
- “SMFJES (FTP server) statement” on page 866
- “SMFLOGN (FTP server) statement” on page 867
- “SMFREN (FTP server) statement” on page 868
- “SMFRETR (FTP server) statement” on page 869
- “SMFSTOR (FTP server) statement”

SMFSTOR (FTP server) statement

Use the SMFSTOR statement to specify SMF recording options for the STOR (STORE) and STOU (STORE UNIQUE) commands

Syntax

```
►►──SMFSTOR──┬─STD─────┬──────────────────────────────────────────────────►◄
             ├─number──┤
             └─TYPE119─┘
```

Parameters

STD
: Indicates that type 118 SMF STORE and STORE UNIQUE records are issued with the standard subtype of 75.

number
: Indicates that type 118 SMF STORE and STORE UNIQUE records are issued with the given record subtype. The valid range is 1 - 255.

TYPE119
: Indicates that type 119 SMF STORE and STORE UNIQUE records are issued (subtype 70).

Examples

Set the type 118 SMF record subtype for STORE and STORE UNIQUE records to 75:

```
SMFSTOR 75
```

To issue type 119 SMF STORE and STORE UNIQUE records:

```
SMFSTOR TYPE119
```

Usage notes

- There is no default value; however, if the SMF statement is coded for type 118 records, the value specified for the SMF statement is used as the default.
- SMFSTOR statements for each record type (118 and 119) function independently of each other.

- If you do not specify the SMFSTOR statement, SMF records for the STORE and STORE UNIQUE commands are still issued if the SMF statement is present. (Type 118 STORE and STORE UNIQUE records have the subtype specified with the SMF statement; type 119 STORE and STORE UNIQUE records are always subtype 70.) If neither the SMF or SMFSTOR statement is specified, no SMF records are collected for the STORE and STORE UNIQUE commands.

Related topics

- "FTP server user exits" on page 700
- "SMF (FTP server) statement" on page 860
- "SMFAPPE (FTP server) statement" on page 862
- "SMFDCFG (FTP server) statement" on page 863
- "SMFDEL (FTP server) statement" on page 864
- "SMFEXIT (FTP server) statement" on page 865
- "SMFJES (FTP server) statement" on page 866
- "SMFLOGN (FTP server) statement" on page 867
- "SMFREN (FTP server) statement" on page 868
- "SMFRETR (FTP server) statement" on page 869
- "SMFSQL (FTP server) statement" on page 870

SOCKSCONFIGFILE (FTP client) statement

Use the SOCKSCONFIGFILE statement to identify the SOCKS server configuration file the FTP client uses to determine which FTP servers require SOCKS protocols.

Syntax

```
►►──SOCKSCONFIGFILE──file-path──────────────────────────────────────────►◄
```

Parameters

file-path
The z/OS UNIX absolute pathname or the fully qualified MVS data set name of the SOCKS configuration file. In accordance with the convention for absolute pathnames, a z/OS UNIX pathname must begin with a slash (/) character. Any file path not beginning with a slash character is considered a fully qualified MVS data set name.

Examples

To direct the client to use the file /etc/ftp/socks.conf for the SOCKS server configuration, specify the following code:

```
SOCKSCONFIGFILE /etc/ftp/socks.conf
```

To direct the client to use the data set 'socks.config' for the SOCKS server configuration, specify one of the following code:

```
SOCKSCONFIGFILE socks.config
SOCKSCONFIGFILE 'socks.config'
```

Usage notes

- If no SOCKSCONFIGFILE statement is specified, the client does not use SOCKS protocols during connection establishment.
- If the client is connecting to an IPv6 node, the client does not use SOCKS protocols during connection establishment.
- The server ignores the SOCKSCONFIGFILE statement.

Related topic

- "SOCKS configuration statements in SOCKSCONFIGFILE" on page 898

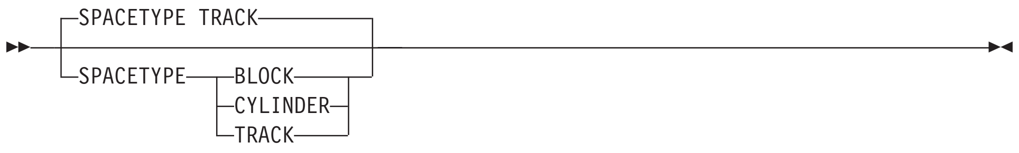
SPACETYPE (FTP client and server) statement

Use the SPACETYPE statement to specify whether newly allocated data sets are allocated in blocks, cylinders, or tracks.

Server This setting applies when creating files on the server's system.

Client This setting applies when creating files on the client's system.

Syntax



Parameters

BLOCK
Use blocks when allocating new data sets.

CYLINDER
Use cylinders when allocating new data sets.

TRACK
Use tracks when allocating new data sets. This is the default.

Examples

Allocate data sets in tracks:

```
SPACETYPE TRACK
```

Usage notes

If you do not supply values on the PRIMARY and SECONDARY statements in order to use the SMS data class, the value on the SPACETYPE statement is ignored and SMS determines the spacetype.

Related topics

- "DATACLASS (FTP client and server) statement" on page 770
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- "PRIMARY (FTP client and server) statement" on page 830

- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- “SECONDARY (FTP client and server) statement” on page 845

SPREAD (FTP client and server) statement

Use the SPREAD statement to specify whether or not the output is in spreadsheet format when the file type is SQL.

Server This setting applies when format is output from the server.

Client This setting applies when format is output from the client.

Syntax

```
     ┌─SPREAD FALSE──────┐
►►───┼───────────────────┼────────────────────────────────────────────────►◄
     └─SPREAD───┬─TRUE──┬┘
                └─FALSE─┘
```

Parameters

TRUE
  Specifies the output is in spreadsheet format.

FALSE
  Specifies the output is not in spreadsheet format. This is the default.

Examples

Format the output to spreadsheet format:

```
SPREAD TRUE
```

Related topics

- “DB2 (FTP client and server) statement” on page 774
- “DB2PLAN (FTP cilent and server) statement” on page 775
- “FILETYPE (FTP client and server) statement” on page 796
- “SQLCOL (FTP client and server) statement”

SQLCOL (FTP client and server) statement

Use the SQLCOL statement to specify the column headings of the output file when FILETYPE is SQL.

Server This setting applies when format is output from the server.

Client This setting applies when format is output from the client.

Syntax

```
   ┌─SQLCOL NAMES──────────┐
►►─┤                       ├──────────────────────────────────────────────────►◄
   └─SQLCOL──┬─ANY────┬────┘
             ├─LABELS─┤
             └─NAMES──┘
```

Parameters

ANY
: Use the label, but if there is no label, the name becomes the column heading.

LABELS
: Use the label of the column headings. If any of the columns do not have labels, the server uses COL*number*, where *number* is the column number reading left to right.

NAMES
: Use the name of the column headings and ignore the labels. This is the default.

Examples

Use the label of the column headings:

```
SQLCOL LABELS
```

Related topics

- "DB2 (FTP client and server) statement" on page 774
- "DB2PLAN (FTP cilent and server) statement" on page 775
- "FILETYPE (FTP client and server) statement" on page 796
- "SPREAD (FTP client and server) statement" on page 874

SSLV3 (FTP client and server connection) statement

Use the SSLV3 statement to enable or disable SSLV3 support for connections that are secured using TLS implemented by FTP (TLSMECHANISM FTP).

Syntax

```
   ┌─SSLV3 FALSE──────────┐
►►─┤                      ├───────────────────────────────────────────────────►◄
   └─SSLV3──┬─FALSE─┬─────┘
            └─TRUE──┘
```

Parameters

FALSE
: Specifies that SSLV3 is disabled. This is the default.

TRUE
: Specifies that SSLV3 is enabled.

Examples

To enable SSLV3 support for connections that are secured using TLS implemented by FTP, code the following statement:

```
SSLV3 TRUE
```

Usage notes

SSLV3 is honored only when TLSMECHANISM FTP is specified.

STARTDIRECTORY (FTP server) statement

Use the STARTDIRECTORY statement to specify which file system is initially used when a new user logs in.

Syntax

```
     ┌─STARTDIRECTORY MVS──────┐
►►───┤                         ├───────────────────────────────────────────►◄
     └─STARTDIRECTORY───┬─HFS─┬┘
                        └─MVS─┘
```

Parameters

HFS
: Use the z/OS UNIX hierarchical file system. The initial directory is the user's root directory in the z/OS UNIX file.

MVS
: Use MVS partitioned data sets. The initial data set name has a prefix of the user ID. See initial working directory consideration in z/OS Communications Server: IP User's Guide and Commands for more information.

Examples

Set the initial user directory to the user's root directory in the z/OS UNIX:

```
STARTDIRECTORY HFS
```

Usage notes

The value of STARTDIRECTORY must be compatible with the ANONYMOUSFILEACCESS value when anonymous logins are enabled and ANONYMOUSLEVEL is 3 or greater.

For example, if ANONYMOUSLEVEL is 3, ANONYMOUSFILEACCESS is MVS, and STARTDIRECTORY is z/OS UNIX, anonymous users receive a filetype error when they attempt to log in to FTP. The anonymous login is rejected by the FTP server.

Related topics

- "ANONYMOUSFILEACCESS (FTP server) statement" on page 741
- "ANONYMOUSLEVEL (FTP server) statement" on page 748

STORCLASS (FTP client and server) statement

Use the STORCLASS statement to specify the SMS storage class as defined by your organization for the FTP server.

Server This setting applies when transferring files from the server's system.

Client This setting applies when transferring files from the client's system.

Syntax

```
►►──STORCLASS──class──────────────────────────────────────────────────►◄
```

Parameters

class
The SMS storage class.

Examples

Use the SMS storage class SMSSTOR when allocating new data sets:

```
STORCLASS SMSSTOR
```

Related topics

- "DATACLASS (FTP client and server) statement" on page 770
- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "UNITNAME (FTP client and server) statement" on page 890
- "UCOUNT (FTP client and server) statement" on page 885
- "VOLUME (FTP client and server) statement" on page 895

SUPPRESSIGNOREWARNINGS (FTP client and server) statement

Use the SUPPRESSIGNOREWARNINGS statement to specify whether FTP issues message EZYFT47I each time it ignores a statement coded in FTP.DATA.

Server This setting applies when starting the FTP server.

Client This setting applies when starting the FTP client.

Syntax

```
    ┌─SUPPRESSIGNOREWARNINGS FALSE────────┐
►►──┴─SUPPRESSIGNOREWARNINGS──┬─TRUE──┬───┴───────────────────────────►◄
                              └─FALSE─┘
```

Parameters

TRUE
Specifies that FTP does not issue message EZYFT47I when ignoring statements coded in FTP.DATA.

Guideline: Do not set SUPRESSIGNOREWARNINGS TRUE until you have verified that the statements in your FTP.DATA configuration file are correct.

FALSE
Specifies that FTP issues message EZYFT47I when ignoring statements coded in FTP.DATA. This is the default.

Examples

Suppress message EZYFT47I while processing statements in FTP.DATA:

```
SUPPRESSIGNOREWARNINGS TRUE
```

Usage notes

- SUPPRESSIGNOREWARNINGS affects only statements in FTP.DATA that follow it. Therefore, code SUPPRESSIGNOREWARNINGS TRUE ahead of any statements for which you do not want the EZYFT47I warning.
- You can suppress EZYFT47I for some, but not all, statements in a single FTP.DATA file, by coding more than one SUPPRESSIGNOREWARNINGS statement. Each instance of SUPPRESSIGNOREWARNINGS is respected, so use it multiple times in FTP.DATA to toggle suppression of warning messages on and off.

TAPEREADSTREAM (FTP server) statement

Use the TAPEREADSTREAM statement to specify whether to use a more efficient read path (read as stream) to retrieve tape data sets from the server.

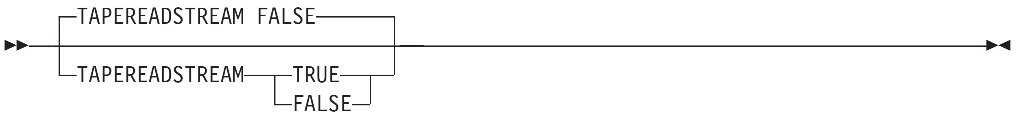
Results: The TAPEREADSTREAM statement takes effect when all of the following conditions are met:

- The file structure is File.
- The transfer mode is Stream.
- One of the following situations is true:
  - The transfer type is E
  - The transfer type is B
  - The transfer type is A and the encoding is SBCS
- The file type is not SQL.

Restrictions: When TAPEREADSTREAM TRUE is configured at the server:

- You cannot retrieve American Standards Association (ASA) tape data sets. The server responds with an error reply if you attempt to retrieve an ASA tape data set.
- You cannot retrieve fixed format tape data sets when TRAILINGBLANKS TRUE is configured. The server responds with an error reply if you attempt to retrieve a fixed format tape data set when TRAILINGBLANKS TRUE is configured.
- If the tape data set contains `<NL>` characters that require translation, the data set format will be incorrect.

Syntax



Parameters

FALSE
Use a common read path for tape data sets. This is the default value.

TRUE
Use a more efficient read path for tape data sets.

Examples

To use a more efficient read path for tape data sets:

```
TAPEREADSTREAM TRUE
```

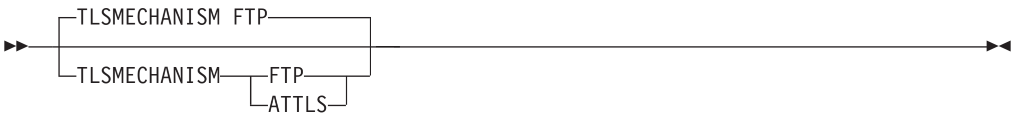
TLSMECHANISM (FTP client and server) statement

Use the TLSMECHANISM statement to specify whether TLS is implemented by AT-TLS or by FTP. AT-TLS is the preferred method for implementing TLS.

Server This setting specifies how TLS security is implemented on the server host. This statement is valid for FTP servers if EXTENSIONS AUTH_TLS is specified.

Client This setting specifies how TLS security is implemented on the client host. This statement is valid for FTP clients if SECURE_MECHANISM TLS or SECURE_MECHANISM SSL is specified.

Syntax



Parameters

FTP
Specifies that secure mechanism TLS is defined by FTP.

Requirement: The KEYRING statement is required if secure mechanism TLS is defined by FTP.

ATTLS
Specifies that secure mechanism TLS is performed by AT-TLS.

Requirement: AT-TLS must be configured in the TCPIP stack. See z/OS Communications Server: IP Configuration Guide for more information.

Restriction: The KEYRING, CIPHERSUITE, and TLSTIMEOUT statements are ignored when using AT-TLS.

Examples

```
TLSMECHANISM FTP
```

Related topics

- “EXTENSIONS (FTP client and server) statement” on page 792
- “SECURE_FTP (FTP client and server) statement” on page 849
- “SECURE_MECHANISM (FTP client) statement” on page 854
- See z/OS Communications Server: IP Configuration Guide for more information about SSL/TLS security, key rings, and certificates and SSL/TLS.

TLSPORT (FTP client and server) statement

Use the TLSPORT statement to set the secure port on which the FTP client or the FTP server implicitly protects the FTP session with TLS.

If you want to use port 990 for unsecured FTP sessions, use this statement to select a different secure port for implicit secure FTP sessions. If you want to disable support for implicit secure FTP, use a value of 0.

Syntax

```
                 ┌─990──┐
►►──TLSPORT──────┼──────┼──────────────────────────────────────────────────►◄
                 └─port─┘
```

Parameters

port
The port number used for implicit secure FTP sessions. The default is 990. The range of valid values is 0 - 65534.

Result: The specification of a TLSPORT does not cause the server to listen on that port, it only specifies that when the port is used it will behave as an implicit TLS port. See “FTP server cataloged procedure (FTPD) parameters” on page 699 for information about how to specify the port for the listener.

Examples

```
TLSPORT 0
```

Related topics

“SECUREIMPLICITZOS (FTP client and server) statement” on page 851

TLSRFCLEVEL (FTP client and server) statement

Use the TLSRFCLEVEL statement to specify the level of RFC 4217 (*Securing FTP with TLS*) that FTP supports. You can also use the locsite subcommand to set this keyword. For information about RFCs, see Appendix C, “Related protocol specifications,” on page 1465.

Server This setting applies when EXTENSIONS AUTH_TLS is coded in the server's FTP.DATA file.

Client This setting applies when SECURE_MECHANISM TLS is coded in the client's FTP.DATA file.

Restrictions:

- FTP supports the TLSPORT statement regardless of the TLSRFCLEVEL setting. FTP connections to the TLSPORT are implicitly secured with TLS as described in the internet draft.
- The TLSRFCLEVEL parameters must be the same on the FTP client and server when using RFC4217 or the CCCNONOTIFY parameter. If the parameters are different, connections might be reset or sessions appear to lock up and eventually timeout.
- The CCCNONOTIFY option is not valid with TLSMECHANISM ATTLS. If both are specified, the use of the CCC command causes the FTP session to fail. If CCCNONOTIFY is required for the partner system, configure TLSMECHANISM FTP with associated statements and exemption in the TTLSRules.

Syntax

```
     ┌─TLSRFCLEVEL DRAFT──────────────────┐
►►───┤                                    ├──────────────────────────────────────────────►◄
     └─TLSRFCLEVEL───┬─DRAFT───────┬──────┘
                     ├─RFC4217─────┤
                     └─CCCNONOTIFY─┘
```

Parameters

DRAFT
Specifies that FTP supports the Internet-draft revision of RFC 4217. This is the level of RFC 4217 support that z/OS FTP has offered since Communications Server V1R2. This is the default.

Guideline: Specify this option, or allow it to default, to maintain the pre-V1R9 support for FTP TLS-protected sessions.

RFC4217
Specifies that FTP supports RFC 4217.

CCCNONOTIFY
Specifies that FTP does not issue the TLSshutdown after sending or receiving the CCC command. RFC 4217 did not mandate this flow until Internet draft revision 14.

Examples

Code this statement in the client's FTP.DATA file to enable RFC 4217 compliance:

```
TLSRFCLEVEL RFC4217
```

Related topics

- "EXTENSIONS (FTP client and server) statement" on page 792
- "SECURE_MECHANISM (FTP client) statement" on page 854
- "TLSPORT (FTP client and server) statement" on page 880
- See z/OS Communications Server: IP Configuration Guide for more information about customizing TLS and Kerberos.

TLSTIMEOUT (FTP client and server) statement

Use the TLSTIMEOUT statement to set a timeout for TLS handshake processing. This timeout is the maximum time between full TLS handshakes. If this time period has not been reached since the last full handshake, a partial handshake occurs when a data connection is protected by TLS.

Server Specifies how often the server requires a full handshake.

Client Specifies how often the client requires a full handshake.

Syntax

```
►►──TLSTIMEOUT──seconds──────────────────────────────────────────────►◄
```

Parameters

seconds
The number of seconds in the range 0 - 86 400. Any value outside of this range reverts to the default of 100.

Examples

```
TLSTIMEOUT 60
```

Related topics

- “EXTENSIONS (FTP client and server) statement” on page 792
- “SECURE_MECHANISM (FTP client) statement” on page 854
- “TLSMECHANISM (FTP client and server) statement” on page 879

TRACE (FTP client and server) statement

Use the TRACE statement to start tracing for FTP.

Server The trace output is written to syslog.

Client The trace output is written to stdout.

Syntax

```
►►──TRACE────────────────────────────────────────────────────────────►◄
```

Parameters

This statement has no parameters.

Examples

To specify that FTP server trace output should be directed to syslog, code the following in the server's FTP.DATA:

```
TRACE
```

Usage notes

- TRACE is equivalent to entering DEBUG BAS or to entering the following four DEBUG statements:

– DEBUG CMD
– DEBUG INT
– DEBUG FSC
– DEBUG SOC

Note that tracing can have a major performance impact on FTP. Consider using the DEBUG statements to request only the kinds of general traces that are needed.

Related topic

- “DEBUG (FTP client and server) statement” on page 778

TRACECAPI (FTP client) statement

Use the TRACECAPI statement to define a control for tracing for a user-written program that uses the FTP client application programming Interface (API) to the z/OS FTP client. This interface is described in z/OS Communications Server: IP Programmer's Guide and Reference.

Syntax

```
   ┌─TRACECAPI CONDITIONAL──────┐
►►─┼────────────────────────────┼──────────────────────────────────────────►◄
   └─TRACECAPI───┬─CONDITIONAL─┬┘
                 ├─ALL─────────┤
                 └─NONE────────┘
```

Parameters

CONDITIONAL
Specifies that tracing by the FTP client API of requests from a user program is conditional. Tracing is based on the setting of the FCAI_TraceIt field prior to issuing the request to the interface. This is the default.

ALL
Specifies that all requests are traced by the FTP client API.

NONE
Specifies that none of the requests are traced by the FTP client API.

Examples

To specify that all requests are traced, use the following code:

```
TRACECAPI  ALL
```

Related topics

For more information about the trace and the interface parameter that the user program uses to control the trace, see the FTP client API information in z/OS Communications Server: IP Programmer's Guide and Reference.

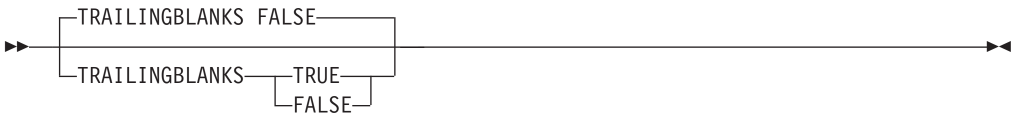
TRAILINGBLANKS (FTP client and server) statement

Use the TRAILINGBLANKS statement to specify whether trailing blanks in a fixed format data set are transferred when the data set is transferred.

Server This setting applies when the server is the sending site.

Client This setting applies when the client is the sending site.

Syntax



Parameters

TRUE
Specifies that the trailing blanks in a fixed format data set are included when the data set is sent.

FALSE
Specifies that the trailing blanks in a fixed format data set are not sent. This is the default.

Examples

Send the fixed format data set and include trailing blanks:

```
TRAILINGBLANKS TRUE
```

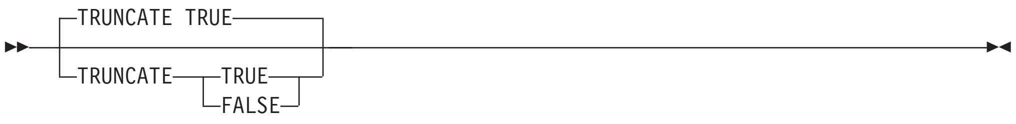
TRUNCATE (FTP client and server) statement

Use the TRUNCATE statement to specify what action should be taken if WRAPRECORD FALSE is specified, and it is determined that an input record is longer than the LRECL of the new file.

Server This setting applies when transferring files to the server's system.

Client This setting applies when transferring files to the client's system.

Syntax



Parameters

TRUE
Specifies that TRUNCATING records is allowed. Even if it is determined that records were truncated, file transfer continues and a warning message is issued when the transfer is complete.

FALSE
Specifies that TRUNCATING records is not allowed. If it is determined that records are truncated, then set an error, and fail the file transfer. If the option WRAPRECORD TRUE is set, the long records are wrapped, not truncated, and no error is set.

Examples

FTP detects a record longer than LRECL, sets an error of 1 003 and fails the transfer of the file.

```
WRAPRECORD TRUE
TRUNCATE FALSE
```

UCOUNT (FTP client and server) statement

Use the UCOUNT statement to set the unit count for new data set allocations.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax

```
►►──UCOUNT──┬────────────┬──────────────────────────────────────────────►◄
            └─unit-count─┘
```

Parameters

unit-count
: The unit count to be specified for new data set allocations. Valid values are 1 - 59 (inclusive), or the letter P for parallel mount requests. UCOUNT has no default value. If you do not specify a UCOUNT value, the FTP server does not specify a unit count for new allocations. The unit count used is the system default.

Examples

To specify a unit count of two, use the following code:

```
UCOUNT 2
```

To specify parallel mounts, use the following code:

```
UCOUNT P
```

Usage notes

- The UCOUNT statement should not be used with an SMS storage class. Any UCOUNT value you specify overrides whatever is specified for the SMS managed dataclass being used.
- UCOUNT can be dynamically modified using the SITE and LOCSITE commands. See z/OS Communications Server: IP User's Guide and Commands for more information about these commands.

Related topics

- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- “STORCLASS (FTP client and server) statement” on page 877

UCSHOSTCS (FTP client and server) statement

Use the UCSHOSTCS statement to specify the EBCDIC code set to be used for data conversion to or from Unicode. If the UCSHOSTCS statement is not used, the current code set for FTP host is used.

Syntax

```
►►──UCSHOSTCS──code_set──────────────────────────────────────────────►◄
```

Parameters

code_set
The EBCDIC code set that is to be used when converting to or from Unicode. See the z/OS XL C/C++ Programming Guide for the valid EBCDIC code set names.

Examples

To set up for conversion between Unicode and IBM 932, use the following code:

```
UCSHOSTCS IBM-932
```

Related topics

- "UCSSUB (FTP client and server) statement"
- "UCSTRUNC (FTP client and server) statement" on page 887

UCSSUB (FTP client and server) statement

Use the UCSSUB statement to specify whether Unicode-to-EBCDIC conversion should use the EBCDIC substitution character or cause the data transfer to be terminated if a Unicode character cannot be converted to a character in the target EBCDIC code set.

Syntax

```
   ┌─UCSSUB FALSE──────┐
►►─┼───────────────────┼──────────────────────────────────────────►◄
   └─UCSSUB──┬─TRUE──┬─┘
             └─FALSE─┘
```

Parameters

TRUE
Specifies that the EBCDIC substitution character is used to replace any Unicode character that cannot successfully be converted. Data transfer continues.

FALSE
Specifies that the data transfer is terminated if any Unicode character cannot be successfully converted.

Examples

To specify that data transfer should be terminated if unicode translation is unsuccessful, use the following code:

```
UCSSUB FALSE
```

Related topics

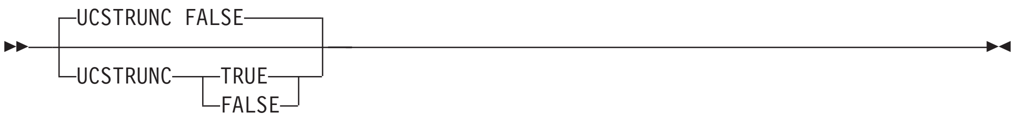
- "UCSHOSTCS (FTP client and server) statement" on page 886
- "UCSTRUNC (FTP client and server) statement"

UCSTRUNC (FTP client and server) statement

Use the UCSTRUNC statement to specify whether the transfer of Unicode data should be aborted if truncation occurs at the MVS host. Truncation can occur if the LRECL of the receiving data set is not large enough to contain a line of Unicode data after it has been converted to EBCDIC.

UCSTRUNC applies to inbound data transfers only.

Syntax



Parameters

TRUE
: Specifies that truncation is allowed. The data transfer continues even if EBCDIC data is truncated.

FALSE
: Specifies that truncation is not allowed. The transfer is to be aborted if the LRECL of the receiving data set is too small to contain the data after conversion to EBCDIC.

  Result: The setting of CONDDISP determines what happens to the target data set if the transfer is aborted.

Examples

To specify that truncation is not allowed, use the following code:

```
UCSTRUNC FALSE
```

Related topics

- "UCSHOSTCS (FTP client and server) statement" on page 886
- "UCSSUB (FTP client and server) statement" on page 886

UMASK (FTP client and server) statement

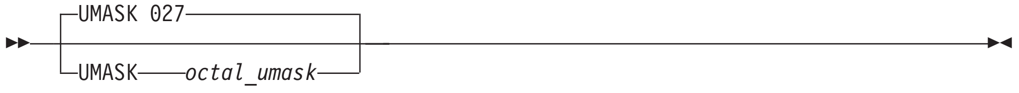
Use the UMASK statement to define the file mode creation mask.

The file mode creation mask defines which permission bits are NOT to be set on when a file is created. When a file is created, the permission bits requested by the file creation are compared to the file mode creation mask, and any bits requested by the file creation that are not allowed by the file mode creation mask are turned off.

Server This setting applies when creating z/OS UNIX files on the server's system.

Client This setting applies when creating z/OS UNIX files sets on the client's system.

Syntax



Parameters

octal_umask
The octal umask.

Examples

When a file is created, the permission bits for file creation are 666 (-rw-rw-rw-). If the file mode creation mask is 027, the requested permissions and the file mode creation mask are compared:

```
110110110   - 666
000010111   - 027
---------
110100000   - 640
```

When the UMASK is set to 027, the actual permission bits set for a file when it is created is 640 (-rw-r-----).

Usage notes

You cannot use FTP to create z/OS UNIX files that have execute permissions. If you require execute permissions, use the **site** and **chmod** commands or **locsite chmod** subcommand after the file is created. For more information about **site** and **locsite**, see z/OS Communications Server: IP User's Guide and Commands.

UNICODEFILESYSTEMBOM (FTP client and server) statement

Use the UNICODEFILESYSTEMBOM statement to specify whether to add a byte order mark (BOM) to a file stored in the local file system when the file system code page is Unicode. You can also use the SIte and LOCSIte subcommands to set this keyword.

Restriction: UTF-8 and UTF-16 are the only Unicode encodings supported in the file system by z/OS FTP.

Result: The BOM stored with the file is determined by the encoding used to store the file rather than by the format of the BOM sent with the file.

Server This setting applies when you are storing Unicode data into the server's file system.

Client This setting applies when you are storing files as Unicode on the client's file system.

Syntax

```
    ┌─UNICODEFILESYTEMBOM ASIS──────────┐
►►──┤                                   ├─────────────────────────────────────────►◄
    └─UNICODEFILESYTEMBOM──┬─ASIS───┬───┘
                           ├─ALWAYS─┤
                           └─NEVER──┘
```

Parameters

ASIS
If a BOMk is present in a Unicode file that is received from the network, store the file with a BOM. If a BOM is not present, store the file without a BOM. The default is ASIS.

ALWAYS
Always include a BOMwhen storing the file. If the file is received without a BOM, insert a BOM into the file.

NEVER
Never include a BOM when storing a UNICODE file. If the file is received with a BOM, discard it before storing the file.

The UNICODE BOM, U+FEFF, can also be interpreted as *zero width nonbreaking space*. z/OS FTP considers only the first character of the file as a possible BOM. No other instance of the BOM sequence in the file is affected by this setting.

Results:

- When appending to a nonexistent regular z/OS UNIX file or MVS data set, the FTP server abides by the UNICODEFILESYTEMBOM setting.
- When appending to an existing regular z/OS UNIX file or MVS data set, the FTP server always strips a leading BOM from the incoming file. This prevents a superfluous BOM from being inserted in the middle of the server file.
- When storing or appending to a z/OS UNIX named pipe, the FTP server always applies the UNICODEFILESYSTEMBOM setting. Multiple transfers into the same named pipe can result in multiple BOM byte sequences inserted into the named pipe.

Guidelines:

- The presence or absence of a BOM can affect applications that process UNICODE files. Consult documentation for applications that process your files or data sets.
- Do not use a BOM when storing UNIX system services configuration files.
- Multiple transfers into a z/OS UNIX named pipe can result in multiple BOM byte sequences being inserted into the named pipe when the UNICODEFILESYSTEMBOM value is ASIS or ALWAYS. To prevent superfluous BOM byte sequences from being inserted in a named pipe, consider setting the UNICODEFILESYSTEMBOM value to NEVER after the first transfer into the named pipe.

Examples

To transfer a UTF-8 file to the server, save it in the server file system as UTF-8, and to guarantee the destination file contains a Byte Order Mark, code the following statements in the server's FTP.DATA:

```
ENCODING MBCS
MBDATACONN(UTF-8,UTF-8)
UNICODEFILESYSTEMBOM ALWAYS
```

Related topics

- "MBDATACONN (FTP client and server) statement" on page 815
- "UNIXFILETYPE (FTP client and server) statement" on page 891

UNITNAME (FTP client and server) statement

Use the UNITNAME statement to specify the unit type for allocation of new data sets.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax

```
►►──UNITNAME──┬──────┬──────────────────────────────────────────────►◄
              └─type─┘
```

Parameters

type
  The type of either direct access or tape devices.

SYSDA
  If *type* is not specified, the unit type used for allocation is the system default.

Examples

- Set the unit type for new data sets to 3380:
  `UNITNAME 3380`
- Set the unit type for new data sets to TAPE:
  `UNITNAME TAPE`

Usage notes

- If you do not use the UNITNAME statement to specify the *type*, the unit type used for allocation is the system default unit.
- If the STORCLASS statement is also specified, the SMS storage class might contain settings that override the UNITNAME *type*.
- It is preferable that you do not use the UNITNAME statement if you are using an SMS storage class.
- The UNITNAME can name a dynamic device.

Related topics

- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "STORCLASS (FTP client and server) statement" on page 877

UNIXFILETYPE (FTP client and server) statement

Use the UNIXFILETYPE statement in the FTP server and client to indicate whether to treat z/OS UNIX file system files as regular files or as UNIX named pipes during file transfer. The site and locsite subcommands are also available to set this value.

Server This setting applies to files in the server's z/OS UNIX file system when the server is processing the APPE, RETR, and STOR commands.

The server ignores this setting when processing the RNFR, RNTO, and DELE. commands. You can use these commands to rename or delete regular files and named pipes regardless of the UNIXFILETYPE setting.

The server accepts the XFIF (create named pipe) command regardless of the UNIXFILETYPE setting.

When the server is processing LIST and NLST commands to list files in the z/OS UNIX file system, both named pipes and regular files appear regardless of the UNIXFILETYPE setting.

Restrictions:

- You cannot restart a file transfer to a named pipe in the server z/OS UNIX file system.
- The server does not support the STOU command when UNIXFILETYPE is set to FIFO.
- Anonymous users are not allowed to read from or write to named pipes in the server z/OS UNIX file system.

Requirements: When the server file exists before it receives the APPE or STOR command, perform the following actions:

- Set UNIXFILETYPE to FILE when the file is a regular file.
- Set UNIXFILETYPE to FIFO when the file is a named pipe.

Client This setting applies to files in the client's z/OS UNIX file system when the client is processing the following subcommands:

- APpend
- Get
- MGet
- MPut
- PUt

Restriction: You cannot restart a file transfer to a named pipe in the client z/OS UNIX file system.

Syntax

```
   ┌─UNIXFILETYPE FILE─────┐
►►─┴─UNIXFILETYPE──┬─FIFO─┬┴──────────────────────────────────────────────►◄
                   └─FILE─┘
```

Parameters

FILE
Treat files in the z/OS UNIX file system as regular files for file storage and retrieval. This is the default value.

Result: When FTP stores a file that does not yet exist in the z/OS UNIX file system, FTP stores the file as a regular file.

Requirement: When FTP stores to a file that already exists in the z/OS UNIX file system, the file must be a regular file.

FIFO
Treat files stored in the z/OS UNIX file system as named pipes for file storage and retrieval.

Result:

- When storing a file that does not yet exist in the z/OS UNIX file system, FTP stores the file as a named pipe.
- When storing to a named pipe that already exists in the z/OS UNIX file system, FTP appends the incoming data to the existing data. This is true for both the APPE (append) and STOR (store) commands.

Requirement: When storing to a file that already exists in the z/OS UNIX file system, the file must be a named pipe.

Restrictions:

- You can append only to existing named pipes.
- You cannot restart a transfer into a named pipe.
- The z/OS operating system does not serialize access to named pipes. Multiple processes can read from or write to the same named pipe simultaneously. When a process reads from a named pipe, data is removed from the named pipe. That data is not presented to other processes that read from the same named pipe. When a process writes to a named pipe, the data it writes might appear in the named pipe interleaved with data written by other processes.

Examples

To treat the files in the z/OS UNIX file system as UNIX named pipes for file transfer, use the following code:

```
UNIXFILETYPE FIFO
```

Related topics

- "FIFOOPENTIME (FTP client and server) statement" on page 795
- "FIFOIOTIME (FTP client and server) statement" on page 794

VCOUNT (FTP client and server) statement

Use the VCOUNT statement to set the volume count for new data set allocations when writing to tapes.

Server	This setting applies when creating data sets on the server's system.
Client	This setting applies when creating data sets on the client's system.

Syntax

```
          ┌─59───────────┐
►►──VCOUNT─┴─volume-count─┴──────────────────────────────────────────►◄
```

Parameters

volume-count
Valid values are integers from 1 - 255 (inclusive). The default value is 59.

Examples

To allow multiple volumes for data set allocation, use the following code:

```
VCOUNT 2
VOLUME (WRKLB1,WRKLB2)
```

Usage notes

- VCOUNT can be dynamically modified using the SITE and LOCSITE commands. See z/OS Communications Server: IP User's Guide and Commands for more information about these commands.

Related topics

- See the information about storage management subsystem (SMS) in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "STORCLASS (FTP client and server) statement" on page 877
- "VOLUME (FTP client and server) statement" on page 895

VERIFYUSER (FTP server) statement

Use the VERIFYUSER statement to indicate whether the FTP server should verify that every user ID used to log into FTP has been granted access to the server's port profile in the SERVERAUTH class.

Tips:

- The FTP server port profile is the same profile that is checked for TLS secured sessions when SECURE_LOGIN VERIFY_USER is coded in FTP.DATA. See "SECURE_LOGIN (FTP server) statement" on page 852 for more information.
- When sessions are secured with TLS and VERIFYUSER TRUE is coded in FTP.DATA, the server verifies the user access to the FTP server port profile regardless of the SECURE_LOGIN value.

Syntax

```
    ┌─VERIFYUSER FALSE──────┐
►►──┴─VERIFYUSER──┬─FALSE─┬─┴──────────────────────────────────────────►◄
                  └─TRUE──┘
```

Parameters

TRUE

If the SERVAUTH class is active and a profile has been defined for the FTP port, the connection is allowed only if the user ID has a minimum of READ access to the profile.

The resource name is as follows:

```
EZB.FTP.systemname.ftpdaemonname.PORTxxxx
```

xxxx is replaced by the port number for the FTP daemon. The profile name can contain wildcard values to the extent that the security product allows. All security product rules apply.

For example, if the procedure FTPD is used to start the FTP daemon on system MVS164 and the FTP daemon uses the default FTP port 21, the resource name is:

```
EZB.FTP.MVS164.FTPD1.PORT21
```

To protect all ports with a single profile, you could use the following security product profile name:

```
EZB.FTP.*.FTPD1.PORT*
```

Result: If the VERIFYUSER value is TRUE, but the security product profile is not defined, the FTP server does not verify access to the profile prior to allowing users to log into FTP.

FALSE

The server does not verify access to the profile EZB.FTP.*systemname*.*ftpdaemonname*.PORT*xxxx* before allowing the login.

Restriction: If the session is secured with TLS and SECURE_LOGIN VERIFY_USER is coded in FTP.DATA, the server checks the user's access to the profile as described in "SECURE_LOGIN (FTP server) statement" on page 852 regardless of the VERIFYUSER setting.

Examples

To request that the FTP server verify user access to the SERVAUTH profile for all sessions regardless of whether they are secured with TLS and regardless of whether TLS level 3 authentication is requested, code this statement in FTP.DATA:

```
VERIFYUSER TRUE
```

You should also define the port profile of the server in the SERVAUTH class of your security product.

For example, if the FTPD procedure is used to start the FTP daemon on system MVS164, and the FTP daemon uses the default FTP port 21, the resource name is as follows:

```
EZB.FTP.MVS164.FTPD1.PORT21
```

If all systems use the same access list and generic profile checking is active for the SERVAUTH class, you can use the following profile name:

```
EZB.FTP.*.FTPD1.PORT21
```

To protect all ports with a single profile, you can use the following security product profile name:

```
EZB.FTP.*.FTPD1.PORT*
```

Related topic

- "SECURE_LOGIN (FTP server) statement" on page 852

VOLUME (FTP client and server) statement

Use the VOLUME statement to specify the volume serial number or a list of volume serial numbers for allocation of new data sets.

Server This setting applies when creating data sets on the server's system.

Client This setting applies when creating data sets on the client's system.

Syntax

```
►►──VOLUME──┬─name──────────┬─────────────────────────────────────────────►◄
            └─(serial-list)─┘
```

Parameters

name
The volume serial number.

(serial-list)
A list of volume serial numbers for new data set allocations.

Examples

Use two volumes for new data set allocations:

```
VOLUME (WRKLB2,WRKLB4)
```

Usage notes

- If you do not use the VOLUME statement to specify the *name*, the volume serial number used for allocation is the system default volume list.
- If the STORCLASS statement is also specified, the SMS storage class might contain settings that override the VOLUME *name*.
- It is preferable that you do not use the VOLUME statement if you are using an SMS storage class.
- When transferring a variable-length file to multiple volumes on MVS, only the last file contains the correct DCB characteristics.
- If you specify multiple volumes, specify them in the order you prefer them to be allocated.

Related topics

- See storage management subsystem (SMS) information in z/OS Communications Server: IP Configuration Guide for more information about specifying attributes when allocating new data sets.
- "DSNTYPE (FTP client and server) statement" on page 783
- "EATTR (FTP client and server) statement" on page 788
- "STORCLASS (FTP client and server) statement" on page 877
- "VCOUNT (FTP client and server) statement" on page 892

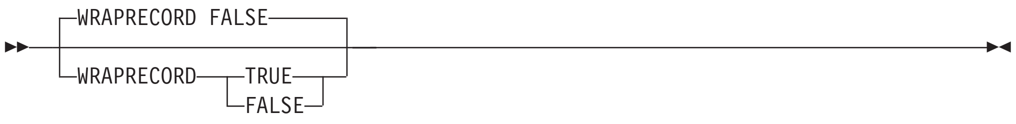
WRAPRECORD (FTP client and server) statement

Use the WRAPRECORD statement to specify how the FTP server or client treats an incoming data record longer than the logical record in which it is to be stored.

Server This setting applies when transferring data sets to the server's system.

Client This setting applies when transferring data sets to the client's system.

Syntax



Parameters

TRUE
Indicates that data is wrapped to the next record if no new-line character is encountered before the logical record length is reached.

FALSE
Indicates that data is truncated if no new-line character is encountered before the logical record length is reached. This is the default. If TRUNCATE is also set to FALSE, an error is set and the file transfer fails.

Examples

Truncate data if no new-line character is encountered before the logical record length is reached:

```
WRAPRECORD FALSE
```

Results

If WRAPRECORD is specified and the data is multibyte characters, it is possible that wrapping will occur in the middle of a multibyte character, making the data unusable by some applications.

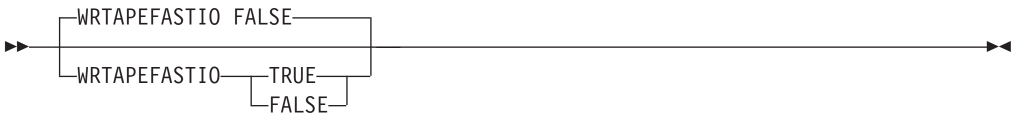
WRTAPEFASTIO (FTP client and server) statement

Use the WRTAPEFASTIO statement to specify whether a write to tape of ASCII data in Stream mode can use the BSAM I/O routine instead of the Language Environment Run-Time Library function *fwrite*().

Server This setting applies when transferring files to the server's system.

Client This setting applies when transferring files to the client's system.

Syntax



Parameters

TRUE
: Indicates that a write to tape of ASCII data in Stream mode is allowed to use the BSAM I/O routine instead of the Language Environment Run-Time Library *fwrite*() function. This allows the data set to be processed without embedded hexadecimal values being interpreted as print control characters.

FALSE
: Indicates that a write to tape of ASCII data in Stream mode must use the Language Environment Run-Time Library *fwrite*() function. This is the default and is used to take advantage of the features of the Language Environment Run-Time Library.

Examples

Allow ASCII Stream data to be written to tape using the BSAM I/O routine:

```
WRTAPEFASTIO TRUE
```

Require ASCII Stream data be written to tape using the Language Environment Run-Time Library:

```
WRTAPEFASTIO FALSE
```

XLATE (FTP server) statement

Use the XLATE statement to specify a data set containing translate tables to be used for the data connection.

Syntax

```
►►──XLATE──name──────────────────────────────────────────────────────►◄
```

Parameters

name
: Specifies a 1- to 8-character name corresponding to a data set that contains translate tables.

  FTP looks first for an environment variable called _FTPXLATE_*name*. If the environment variable exists, its value is used as the data set name.

  Restriction: The environment variable name must be all uppercase, although the XLATE parameter can be in mixed case.

  If the environment variable does not exist, FTP looks for a data set called *hlq*.*name*.TCPXLBIN.

Examples

```
XLATE FRED
```

If environment variable _FTPXLATE_FRED=FREDDYS.TABLES is defined for the FTP server, this statement specifies that the translate tables in data set FREDDYS.TABLES should be used for the data connection.

If there is no such environment variable defined, this statement specifies that the translate tables data set *hlq*.FRED.TCPXLBIN should be used.

Usage notes

- SBDATACONN and XLATE are mutually exclusive statements. If both statements appear in your FTP.DATA file, XLATE is ignored.
- The XLATE statement (and its value) is not case sensitive, but the name of the corresponding environment variable must be all uppercase or FTP does not recognize it.

Related topics

- Appendix A, "Translation tables," on page 1417
- "CCXLATE (FTP server) statement" on page 759
- "CTRLCONN (FTP client and server) statement" on page 769
- To see the search order that determines the conversion for the control connection, see z/OS Communications Server: IP Configuration Guide.

FTP server environment variables

Table 53 provides a list of environment variables used by FTP server that can be tailored to a particular installation.

Table 53. FTP server environment variables

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules (or a link to syntax)
_ICONV_UCS2	FTP	Instructs iconv_open(Y, X) about which type of conversion method to set up when there is a choice between direct conversion from X to Y and indirect X to UCS-2 to Y.	This environment variable has no affect on streams that are based on z/OS UNIX files. You can always read and write 0-byte records in z/OS UNIX files.
RESOLVER_CONFIG	FTP	The resolver configuration data sets or files.	None

SOCKS configuration statements in SOCKSCONFIGFILE

The FTP client uses configuration information in a SOCKS configuration data set or file to determine whether to access a given IPv4 FTP server directly or through a SOCKS server. The name of the SOCKS configuration data set or file is specified by coding the SOCKSCONFIGFILE statement in the client's FTP.DATA file. For more information about the SOCKSCONFIGFILE statement, see "SOCKSCONFIGFILE (FTP client) statement" on page 872.

You can code DIRECT or SOCKD statements in the SOCKSCONFIGFILE. A DIRECT statement instructs the FTP client to access the FTP server without using SOCKS. A SOCKD statement directs the client to use SOCKS protocols and the specified SOCKS server to access the FTP server.

You can include comments in the configuration file or data set. Comment lines should start with a semicolon (;) character. Any data on any line that follows a free-standing semicolon (a semicolon surrounded by at least one space on either side) is considered to be a comment.

The order of statements in the SOCKS configuration is important. The client searches the statements in the order they are coded in the SOCKSCONFIGFILE. The first statement that specifies the target FTP server is applied. Code statements that apply to specific FTP servers first, and a general statement for all other servers last.

The configuration information in the SOCKS configuration file consists of the statements in the following topics.

DIRECT statement

Use the DIRECT statement to instruct the FTP client not to use SOCKS for the destinations that are included in the DIRECT statement.

Syntax

```
►►──DIRECT──┬─IPv4_address address_mask──┬──────────────────────────────────────►◄
            └─IPv4_address/num_mask_bits─┘
```

Parameters

direct
Access the FTP server indicated by this statement without using SOCKS protocols.

IPv4_address
Dotted decimal IPv4 address of the FTP server host, or the dotted decimal IPv4 Network ID of the FTP server network or subnet. The network ID can include subnet bits.

address_mask
Dotted decimal IPv4 subnet mask.

num_mask_bits
An integer in the range 1 - 32 that represents the number of bits, counting from left to right, of the network and subnet portion of the IPv4 address mask.

Examples

The following statements instruct the FTP client not to use SOCKS for connections to any FTP servers in the class A 9.0.0.0 network, nor to connections to the host's loopback address.

```
;
; This is my socks configuration
;
direct 9.0.0.0 255.0.0.0                  ; Internal net
direct 127.0.0.1 255.255.255.255          ; Loopback
```

The following statement directs the FTP server not to use SOCKS for connections to the host's loopback address (num_mask_bits is coded instead of address_mask).

```
;
; This is my socks configuration
;
direct 127.0.0.1/32          ; Loopback
```

Usage notes

- You can code as many DIRECT statements as needed to cover your configuration.
- The FTP client always acts as if the statement direct 0.0.0.0 0.0.0.0 were coded last in the SOCKSCONFIGFILE. This statement applies to every possible FTP server and directs the client not to use SOCKS to access any server not covered by a previous statement. Therefore, the client connects to any FTP server for which no statement has been coded in SOCKSCONFIGFILE without using SOCKS. Also, note that if you code this statement in the SOCKSCONFIGFILE explicitly, any statements you coded after that would be ignored because the client always uses the first statement that applies to the FTP server.

SOCKD statement

Use the SOCKD statement to instruct the FTP client to use a SOCKS server for the destinations that are included in the sockd statement.

Syntax

```
►►──┬─SOCKD──┬──┬─@=SOCKS_srv_IPv4_Addr──┬──┬─IPv4_address IPv4_address_mask─┬──────►◄
    ├─SOCKD4─┤  └─@=SOCKS_srv_host_name─┘  └─IPv4_address/num_mask_bits──────┘
    └─SOCKD5─┘
```

Parameters

SOCKD
: The SOCKS server requires the use of SOCKSv5 protocols.

SOCKD4
: The SOCKS server requires the use of SOCKSv4 protocols.

SOCKD5
: The SOCKS server requires the use of SOCKSv5 protocols.

SOCKS_srv_host_name
: The DNS name of the SOCKS server host.

SOCKS_srv_IPv4_addr
: The dotted decimal IPv4 IP address of the SOCKS server host.

IPv4_address
: Dotted decimal IPv4 address of the FTP server host, or the dotted decimal IPv4 Network ID of the FTP server network or subnet. The network ID can include subnet bits.

IPv4_address_mask
: Dotted decimal IPv4 subnet mask.

num_mask_bits
: An integer between 1 and 32 that represents the number of bits, counting from left to right, of the network and subnet portion of the IPv4 address mask.

Examples

In the following example, the first statement instructs the client to use SOCKS V4 protocols and the SOCKSv4 server at IP address 9.1.2.3 for connections to FTP severs within the class C 192.168.1.0 network. The second statement instructs the client to use SOCKSv5 protocols and the SOCKSv5 server at IP address 9.1.2.4 to access any FTP server not covered by a previous statement.

```
sockd4 @=9.1.2.3 192.168.1.0 255.255.255.0 ; Test net
sockd5 @=9.1.2.4 0.0.0.0 0.0.0.0 ; Anything else
```

Usage notes

- You can code as many SOCKD statements as needed to cover your configuration.
- DIRECT and SOCKD statements can be mixed in any order.

Chapter 19. Trivial file transfer protocol

This topic includes the following information about Trivial file transfer protocol (TFTP):

- "Starting TFTPD as a procedure" on page 905
- "Step for starting the TFTP server" on page 907
- "Step for stopping the TFTP server" on page 907

TFTP is installed in the /usr/lpp/tcpip/sbin/ directory.

The TFTP server uses well-known port 69. The TFTP server has no user authentication. Any client that can connect to port 69 on the server has access to TFTP. If the TFTP server is started without a directory, it allows access to the entire z/OS UNIX. To restrict access to the z/OS UNIX, start the TFTP server with a list of directories.

Following are the parameters used for the **tftpd** command:

-l
Logs all the incoming read and write requests and associated information to the system log. Logged information includes the IP address of the requester, the file requested, and whether the request was successful.

-p port
Uses the specified port. The TFTP server usually receives requests on well-known port 69. You can specify the port on which requests are to be received.

-t timeout
Sets the packet timeout. The TFTP server usually waits 5 seconds before presuming that a transmitted packet has been lost. You can specify a different timeout period in seconds.

-r maxretries
Sets the retry limit. The TFTP server usually limits the number of retransmissions it performs because of lost packets to 5. You can specify a different retry limit.

-c concurrency_limit
Sets the concurrency limit. The TFTP server spawns both threads and processes to handle incoming requests. You can specify the limit for the number of threads that can be concurrently processing requests under a single process. When the limit is exceeded, a new process is spawned to handle requests. The default is 200 threads.

-s maxsegsize
Sets the maximum block size that can be negotiated by the TFTP block size option. The default is 8192.

-f file
Specifies a cache file. You can specify a file containing information about files to be preloaded and cached for transmission. A cache file consists of one or more entries. For clarity, place each entry on a separate line. An entry has the form:

```
a | b  <pathname>
```

where:

- *a* indicates that the specified file is cached in ASCII form. The file is preconverted to NETASCII format.
- *b* indicates that the specified file is cached in binary form, with no conversion.

Following are examples of cache file entries:

```
a  /usr/local/textfile
b  local/binaryfile
```

If a relative pathname to the file is specified, the TFTP server searches the specified directories for the file.

The cached version of a file is only used for requests requiring the specified format. For example, the binary cached version of a file is not used in satisfying a request for the file in netascii format. If a file is to be retrieved in both binary and ASCII formats, the user must specify that two copies of the file be cached with one in binary format, and the other in NETASCII format.

Caching is not dynamic. The cache files are read in when the TFTP server is started and are not updated, even if the file on disk is updated. To update or refresh the cache, the TFTP server must be recycled.

-a archive directory
Specifies an archive directory. The files in this directory and its subdirectories are treated as binary files for downloading. This option is useful on EBCDIC machines that act as file servers for ASCII clients. Multiple -a options can be specified; one directory per -a option. Directories must be specified as absolute pathnames. You can specify no more than 20 directories.

directory
Specifies an absolute pathname for a directory. You can specify no more than 20 directories on the tftpd command line.

If the TFTP server is started without a list of directories, all mounted directories are considered active.

If a list of directories is specified, only those specified directories are active. That list is used as a search path for incoming requests that specify a relative pathname for a file.

Activating a directory activates all of its subdirectories.

For a file to be readable by the TFTP server, the file must be in an active directory and have world (other) read access enabled. For a file to be writable by the TFTP server, the file must already exist in an active directory and have world (other) write access.

-b IP address
Uses the specified IP address. The TFTP server usually binds to *in6addr_any* or *inaddr_any*. You can specify the IP address on which requests are to be received. TFTP requests that come in on other IP addresses are not accepted by this instance of TFTPD.

The TFTP server preforks a child process to handle incoming requests when the concurrency limit is exceeded. Consequently, immediately after starting the TFTP server, two TFTP processes exist.

In case of a flood of concurrent TFTP requests, the TFTP server might fork additional processes. When the number of concurrent requests being processed drops below the concurrency limit, the number of TFTP processes is decreased back to two.

Starting TFTPD as a procedure

The following sample [shipped as SEZAINST(TFTPD)] shows how to start TFTPD as a procedure:

```
//TFTPSD   PROC
//*
//* Communications Server IP
//* SMP/E distribution name: EZATTFDP
//*
//*   5694-A01 5655-HAL (C) Copyright IBM Corp. 1997, 2004.
//*   Licensed Materials - Property of IBM
//*   This product contains "Restricted Materials of IBM"
//*   All rights reserved.
//*   US Government Users Restricted Rights -
//*   Use, duplication or disclosure restricted by
//*   GSA ADP Schedule Contract with IBM Corp.
//*   See IBM Copyright Instructions.
//*
//* Function: Trivial File Transfer Protocol Server start
//*
//* Please note:
//*
//* -a  Specify an archive directory.  TFTPD treats files in this
//*     directory and its subdirectories as binary files for uploads
//*     and downloads, regardless of how they were requested by the
//*     client.  Use this option on EBCDIC machines that act as file
//*     servers for ASCII clients.
//*
//*     You can specify up to 20 -a options, one directory per -a
//*     option.  You must specify directories as absolute pathnames.
//*
//* -c  the number of threads to use concurrently.  Make the number
//*     a some reasonable number like 20 and not the default, which
//*     is 200.
//*
//* -t  Set the packet timeout. The TFTP server usually waits 5 seconds
//*     before presuming that a transmitted packet has been lost. You can
//*     specify a different timeout period in seconds.
//*
//* -l  Log all incoming read and write requests and associated
//*     information to the system log. Logged information includes the IP
//*     address of the requestor, the file requested, and whether the
//*     request was successful.
//*
//* -p  Specify the port. The TFTP server usually receives requests on
//*     well-known port 69. You can specify the port in which requests
//*     are to be received.
//*
//* -r  Set the retry limit. The TFTP server usually limits the number
//*     of retransmissions it performs due to lost packet to 5. You
//*     can specify a different retry limit.
//*
//* -s  Set the maximum block size that can be negotiated by the
//*     TFTP block size option. The default is 8192.
//*
//* -f  Secify a cache file. You can specify a file containing
//*     information on files to be preloaded and cached for transmission.
//*     A cache file consists of one or more entries. For clarity, place
//*     each entry on a separate line. See the IP Configuration Guide for
//*     details on this option.
//*
//* -b  Specify IP address. The TFTP server usually binds to in6addr_any
//*     or inaddr_any. You can specify the IP address on which requests
//*     are to be received.
//*     TFTP requests that come in on other IP addresses will not be
//*     accepted by this instance of TFTPD.
//*
//TFTP     EXEC PGM=TFTPD,REGION=0K,TIME=NOLIMIT,
//             PARM='POSIX(ON),ALL31(ON),TRAP(OFF)/
//             -c 20 -t 300'
//*STEPLIB DD DISP=SHR,DSN=TCP.SEZALOAD,
//*           VOL=SER=,UNIT=
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//         PEND
```

Figure 29. TFTPD sample [shipped as SEZAINST(TFTPD)]

Step for starting the TFTP server

To start the TFTP server from the z/OS command line, type the **tftpd** command.

```
tftpd [-l] [-p port] [-t timeout] [-r maxretries] [-c concurrency_limit]
   [-s maxsegsize] [-f file] [-a archive directory [-a ...]]
   [directory ...] [-b IP address]
```

Step for stopping the TFTP server

This topic describes the steps of stopping the TFTP server.

Procedure

Perform the following step to terminate the TFTP server.

Send a SIGTERM signal to the oldest existing TFTP process.

Note: This is the process that has a parent process ID of 1.

Results

When you are done, you have terminated this process, which causes all of its children to terminate.

Chapter 20. Syslog daemon

This topic contains the following information:

Syslog daemon files

The syslog daemon (syslogd) uses the following files:

/dev/console
: Operator console

/dev/operlog
: operlog log stream

/etc/syslog.pid
: The syslogd cataloged procedure stores its process ID in this file when running in normal or local-only mode

/etc/syslog_net.pid
: syslogd stores its process ID in this file when running in the network-only mode

/etc/syslog.conf
: Default configuration file

/dev/log
: Default log path for UNIX datagram socket

/usr/sbin/syslogd
: Symbolic link to the server executable

Starting syslogd with a cataloged procedure

Update the cataloged procedure, syslogd, by copying the sample in SEZAINST(SYSLOGD) to your system or recognized PROCLIB. Specify syslogd parameters and change the data set names to suit your local configuration. See the syslog daemon section of SEZAINST(EZARACF) for SAF considerations for started procedures. When you start syslogd from a procedure that does not use BPXBATCH, the resulting job name is the same as the procedure name. When you start syslogd from the z/OS UNIX shell or from a procedure that uses BPXBATCH, the resulting job name is the user ID or the value of the _BPX_JOBNAME environment variable.

See "Starting syslogd from the UNIX shell" on page 911 for the syntax of the start options for syslogd.

Below is a copy of the sample procedure:

```
//SYSLOGD PROC
//*********************************************************************
//*     Descriptive Name:           SYSLOGD Start Procedure          *
//*                                                                   *
//*     File Name:                  tcpip.SEZAINST(EZASYSLG)          *
//*                                 tcpip.SEZAINST(SYSLOGD)           *
//*                                                                   *
//*     SMP/E Distribution Name:    EZASYSLG                          *
//*                                                                   *
//*     Licensed Materials - Property of IBM                          *
//*     "Restricted Materials of IBM"                                 *
//*     5650-ZOS                                                      *
//*     Copyright IBM Corp. 1992, 2013                                *
//*     Status = CSV2R1                                              *
//*                                                                   *
//* Note:                                                             *
//*  The SYSLOGD Daemon can read its configuration file from either a *
//*  PDS or the HFS. The procedure defaults to the HFS.               *
//*  If you are running the IVP for SYSLOGD or if you simply prefer   *
//*  to use a PDS to store your configuration file,                   *
//*  either delete or comment the CONFHFS DD card                     *
//*  then uncomment the CONFPDS DD card and specify the data set and  *
//*  member name.                                                     *
//*                                                                   *
//*  If you would like to run two instances of syslogd, make a second *
//*  copy of this proc and replace -i with -n in the second instance. *
//*  The instance using -n will process only log messages received    *
//*  over the well-known syslogd port via UDP. One instance must      *
//*  use -i and the other must use -n in order to run two instances.  *
//*                                                                   *
//*  The -c command-line option specifies that syslogd should create  *
//*  any log files or directories which do not already exist.         *
//*                                                                   *
//*  The -i command-line option specifies that syslogd should not     *
//*  process log messages sent to the well-known syslog port via UDP. *
//*********************************************************************
//CONFHFS EXEC PGM=SYSLOGD,REGION=4096K,TIME=NOLIMIT,
//         PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/-c -i'
//*
//*** Examples for specifying environment variables and parameters
//*** (parameters must extend to column 71 and be continued in
//*** column 16):
//*
//*  Example 1: Environment variables inline, MVS config data set
//*
//*CONFPDS EXEC PGM=SYSLOGD,REGION=4096K,TIME=NOLIMIT,
//*        PARM='ENVAR("TZ=EST5EDT")/-c -i -f //''TCPIP.TCPPARMS(SYSLOG
//*            )'''
//*
//*  Example 2: Environment variables inline, UNIX config file
//*
//*CONFHFS EXEC PGM=SYSLOGD,REGION=4096K,TIME=NOLIMIT,
//*        PARM='ENVAR("TZ=EST5EDT")/-c -i -f /user1/syslogd.conf'
//*
//*  Example 3: Environment variables in STDENV DD
//*
//*CONFSTD EXEC PGM=SYSLOGD,REGION=4096K,TIME=NOLIMIT,
//*        PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/-c -i'
//*
//*   For this method, the STDENV DD statement below must be
//*   changed to point to a MVS data set or UNIX file containing
//*   settings for any environment variables.  For example, it should
//*   contain at least TZ (unless you choose to specify TZ in a
//*   different fashion), but can contain other environment variables
//*   as in this example:
//*
//*     SYSLOGD_CODEPAGE=IBM-1047
//*     SYSLOGD_CONFIG_FILE=/user1/syslogd2.conf
//*     SYSLOGD_DEBUG_LEVEL=127
//*     TZ=EST5EDT
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//*   The use of the STDENV DD statement works well when more than
//*   one environment variable is specified, as there is a JCL limit
//*   of 100 characters on the PARM statement.
//*
//*   Note: Language Environment recommends a variable record format
//*         for the STDENV file.
//*
//*   You can also set the TZ environment variable for all applications
//*   in the CEEPRMxx PARMLIB member.  You should define the TZ
//*   environment variable for all three LE option sets (CEEDOPT,
//*   CEECOPT, and CELQDOPT).  For example:
//*
//*     CEECOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*     CEEDOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*     CELQDOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*
//*   For more information on specifying run-time options, see z/OS
//*   Language Environment Programming Guide. For details on setting
//*   the LIBPATH and TZ environment variables, see z/OS UNIX System
//*   Services Command Reference.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.SYSLOGD.ENV(SYSLOGD),DISP=SHR
//* Sample UNIX file containing environment variables:
//*STDENV   DD PATH='/etc/syslogd.env',PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

Figure 30. Syslogd sample cataloged procedure

Starting syslogd from the UNIX shell

Start syslogd from the UNIX shell using the syntax described in this topic.

syslogd [-D *value***][-F** *value***[-f** *conffile***] [-m** *markinterval***] [-p** *logpath***] [-c] [-d] [-i] [-n][-x][-u] [-?]&**

Rules:

- You must start syslogd as a background shell command by specifying an ampersand (&) as the last character on the command. If you do not specify the ampersand (&), control does not return to the shell until syslogd ends. This is especially important if syslogd is started from a shell script, such as /etc/rc.

- If you start syslogd from a cataloged procedure that uses BPXBATCH, you need to include a sleep command in your script after you start syslogd. The sleep command gives syslogd time to initialize before the shell script ends. For more information about including a sleep command, see Starting daemons in z/OS UNIX System Services Planning.
- If you start syslogd from a cataloged procedure that uses BPXBATCH, you need to use a z/OS UNIX file for the STDOUT DD JCL statement and the STDERR DD JCL statement; otherwise, the job will not end.

syslogd recognizes the following options:

-c
Create log files and directories automatically.

-d
Run syslogd in debugging mode.

-D
Specify the default access permissions (modes) to be used by syslogd when creating directories. This parameter is valid only when specified in conjunction with the **-c** option. The parameter value is specified as an octal number 1 - 4 characters in length. Leading zeros can be omitted. The following values can be ORed together to form the parameter value:

2000	Set-GID
1000	Sticky Bit (deletion restricted to owner or superuser)
0400	User read
0200	User write
0100	User list directory
0040	Group read
0020	Group write
0010	Group list directory
0004	Other read
0002	Other write
0001	Other list directory

If the -D option is not specified, the default value 700 is used. Bits other than the bits shown are not valid and are set to 0. For example, you cannot set the set-UID bit for a directory.

-f
Configuration file name.

-F
Specify the default access permissions (modes) to be used by syslogd when creating log files. This parameter is valid only when specified in conjunction with the **-c** option. The parameter value is specified as an octal number 1 - 4 characters in length. Leading zeros can be omitted. The following values can be ORed together to form the parameter value:

0400	User read
0200	User write
0040	Group read
0020	Group write
0004	Other read
0002	Other write

If the -F option is not specified, the default value 600 is used. The actual permissions are modified by the syslogd process's umask value at the time the file is created. This parameter is used only when syslogd must create a log file dynamically; it has no effect on log files that already exist. Bits other than the bits shown are not valid and are set to 0. For example, you cannot set the execute bits, set-UID, set-GID, or the sticky bit for log files.

-i Start in local-only mode. Do not receive messages from the IP network. Messages can be sent by syslogd to remote syslogd instances when running in local-only mode.

-m Number of minutes between mark messages. The default value is 20 minutes.

-n Start in network-only mode. Process messages from the IP network only (UDP datagram socket). Messages can be sent by syslogd to remote syslogd instances when running in network-only mode.

-p Pathname of z/OS UNIX character device for the datagram socket. The default value is /dev/log.

Guideline: This option is not used frequently. If you incorrectly use the **-p** option, syslogd does not function properly.

-u For records received over the AF_UNIX socket (most messages generated on the local system), include the user ID and job name in the record. In this case, a forward slash (/), the user ID, and the job name follows the local host name for messages received over the AF_UNIX socket. The forward slash, which immediately follows the local host name, can be used to determine whether or not the user ID and job name are being recorded. If not recorded, a blank immediately follows the local host name. When user ID or job name is not available, N/A is written in the corresponding field.

-x Do not perform IP address-to-host name resolution for messages received from the IP network.

-? Show syslogd options.

Syslogd environment variables

Table 54 provides a list of environment variables used by syslogd that can be tailored to a particular installation.

Table 54. Syslogd environment variables

Environment variable	Description	Specific coding rules
SYSLOGD_CODEPAGE	Used by the syslog daemon to specify the EBCDIC code page to be used for the configuration file. The default code page is IBM-1047.	The following code pages are supported: • IBM-037 • IBM-273 • IBM-274 • IBM-275 • IBM-277 • IBM-278 • IBM-280 • IBM-281 • IBM-282 • IBM-284 • IBM-285 • IBM-297 • IBM-500 • IBM-871 • IBM-1047 • IBM-1140 • IBM-1141 • IBM-1142 • IBM-1143 • IBM-1144 • IBM-1145 • IBM-1146 • IBM-1147 • IBM-1148 • IBM-1149 Example: `SYSLOGD_CODEPAGE=IBM-1141`
SYSLOGD_CONFIG_FILE	Specifies the name of the syslogd configuration file.	The **-f** start option overrides this value. Example: `SYSLOGD_CONFIG_FILE=/etc/syslog.conf`

SYSLOGD_DEBUG_DATASET	If this environment variable is not set, the debug output goes to STDOUT. The environment variable can be set to STDOUT, which will also cause the output to be written to STDOUT. On SYSLOGD shutdown, the SYSLOGD_DEBUG_VARIABLE is reset, which causes residual debug output to be written to STDOUT.	Write debug output to an MVS dataset
		`export`
		`SYSLOGD_DEBUG_DATASET="//'USER1.SYSLOGD.DEBUG'"`
		Write debug output to a UNIX file
		`export`
		`SYSLOGD_DEBUG_DATASET="/tmp/syslogd_debug"`

Table 54. Syslogd environment variables (continued)

Environment variable	Description	Specific coding rules
SYSLOGD_DEBUG_LEVEL	Specifies the debug level to be used by syslogd.	You can specify the following debug levels. You can add these together in any combination to select the type of debug messages to be written. **1** Base debugging information. **2** Configuration file processing. **4** Processing of messages being logged by syslogd. **8** Automatic archive processing. **16** Operator command processing. **32** Thread-specific processing. **64** Mutex lock processing. Locks that are specific to threads are logged only if the debug level includes 32. For example, `SYSLOGD_DEBUG_LEVEL=91` includes all debugging information except for message handling and thread-specific processing (including locks). The default debug level is 127, which includes all debug information.
SYSLOGD_PATH_NAME	Specifies the path name for the datagram socket.	The **-p** start option overrides this value.

- When starting syslogd from a shell script, export the environment variables before starting syslogd. The following example defines the syslogd configuration file:

```
#
# Shell script to start syslogd
#
export _BPX_JOBNAME='SYSLOGD1'
export SYSLOGD_CONFIG_FILE="//'HLQ.SYSLOGD.CONFIG(DEFAULT)'"
 /usr/sbin/syslogd &
```

- When starting syslogd directly from a started procedure, place the syslogd environment variables in a z/OS UNIX file or MVS data set. Use the following technique to pass the environment variables to syslogd.

```
// PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'

//STDENV DD PATH='/etc/syslogd.env',PATHOPTS=(ORDONLY)
or
//STDENV DD DSN=HLQ.SYSLOGD.ENV(DEFAULT),DISP=SHR
```

  When you use an MVS data set for your syslogd environment variables, place the environment variables in a data set with the VB record format [RECFM(VB)] and a logical record length of 256 [LRECL(256)]. If you use any other record format for the data set, use the _CEE_ENVFILE_S environment variable in place of the _CEE_ENVFILE environment variable in your syslogd started procedure. When the _CEE_ENVFILE_S environment variable is used, the system removes trailing blank spaces from each *NAME=VALUE* line that is read. For additional information about the _CEE_ENVFILE_S environment variable, see z/OS XL C/C++ Programming Guide.

Syslogd configuration statements

There are two types of configuration information. The first type is a set of global parameters that you use to configure some operational aspects of syslogd. The second type is a set of rules that you use to define the mapping between the information that is logged to syslogd and the destination of that information.

A sample configuration file is included in /usr/lpp/tcpip/samples/syslog.conf.

Global syslogd configuration statements

This topic contains descriptions of the global syslogd configuration statements.

ArchiveCheckInterval statement

Use the ArchiveCheckInterval statement to specify the interval of time at which syslogd checks UNIX file system utilization. Each configured rule that specifies a UNIX file as the destination and that specifies the -N parameter to indicate that it is eligible for archive processing is a candidate for automatic archival. If you also configure the ArchiveThreshold statement with a nonzero value, the set of UNIX file systems that contain the candidate UNIX files is checked for the percentage used to determine whether threshold archiving needs to be performed.

If the ArchiveCheckInterval statement is specified multiple times, syslogd uses the last instance of the statement.

```
►►──ArchiveCheckInterval──minutes──────────────────────────────────────────►◄
```

minutes
  Specifies the interval at which syslogd checks file system utilization, in minutes. Valid values range 1 - 1 440. If you do not specify this statement, the default is 10 minutes.

ArchiveThreshold statement

Use the ArchiveThreshold statement to specify a percentage of UNIX file system utilization. Each configured rule that specifies a UNIX file as the destination and that specifies the -N parameter to indicate that it is eligible for archive processing is a candidate for automatic archival. When you specify this statement with a nonzero value, each UNIX file system that contains one or more candidate UNIX files is checked at the interval that is configured or the default value to determine the percentage of the file system being used. If the percentage of file use exceeds the specified threshold, syslogd automatically archives a set of UNIX files until the percentage of file use is below the minimum threshold. The minimum threshold is 50 percent of the value that you configure on this statement. For example, if you specify an archive threshold of 80 percent, syslogd archives UNIX files until the percentage of file use is below 40 percent. Syslogd archives files from the largest to the smallest.

If the ArchiveThreshold statement is specified multiple times, syslogd uses the last instance of the statement.

You can use this statement to perform archiving when one or more file systems reach a configured threshold, or you can use the ArchiveTimeOfDay statement to perform archiving at a specific time of day. You can also specify both statements.

Requirement: The **-c** start option is required when you use the ArchiveThreshold statement.

```
►►──ArchiveThreshold──percentage───────────────────────────────────────────►◄
```

percentage
  Specifies the percentage value of UNIX file system use that triggers automatic

archival. Valid values for *percentage* are in the range 0 - 99. If you do not specify this statement, the default is 70. If you specify 0, syslogd does not perform automatic threshold archiving.

ArchiveTimeOfDay statement

Use the ArchiveTimeOfDay statement to specify a local time of day to perform an archive of all eligible UNIX files. Each configured rule that specifies a UNIX file as the destination and that specifies the -N parameter to indicate that it is eligible for archive processing is a candidate for automatic archival.

You can use this statement to perform archiving at a specific local time of day, or you can use the ArchiveThreshold statement to perform archiving when one or more file systems reaches a configured threshold. You can also specify both statements.

If the ArchiveTimeOfDay statement is specified multiple times, syslogd uses the last instance of the statement.

Requirement: The **-c** start option is required when you use the ArchiveTimeOfDay statement.

```
►►──ArchiveTimeOfDay──time──────────────────────────────────────────►◄
```

time
: Specifies the local time of day to perform an automatic archival of all UNIX files. Specify the local time value in hours and minutes, using a 24 hour clock. For example, the value 00:01 means to archive at 1 minute past midnight. There is no default; if you do not specify this statement, syslogd does not perform automatic time of day archival.

BeginArchiveParms statement

Use the BeginArchiveParms statement to specify the prefix and allocation information for the archive destination data set. UNIX files are archived only to MVS data sets when the automatic archive function is used. The complete archive data set name is composed of several components, depending on the type of destination data set that is being used. Use the -N parameter on a specific syslogd rule to specify the type of destination data set.

You should use this method of archiving files only if you do not offload files using the provided sample configuration and procedure. See z/OS Communications Server: IP Configuration Guide for more information about offloading log files. Because both of these methods rely on creating new log files, results could be unpredictable if you try to use both methods together.

When the destination is a Generation Data Group (GDG) data set, the complete data set name is:

prefix.qualifier.gdg_suffix where:

- *prefix* is the value specified on the BeginArchiveParms statement.
- *qualifier* is the value specified on the -N parameter on a specific syslogd rule.
- *gdg_suffix* is the value automatically supplied for GDG data sets to make them unique.

The following example shows a GDG data set name:

```
USER1.SYSARCH.TRACE.G0007V00
```

When the destination is a sequential data set, the complete data set name is:

prefix.qualifier.date_suffix.time_suffix where:
- *prefix* is the value specified on the BeginArchiveParms statement.
- *qualifier* is the value specified on the -N parameter on a specific syslogd rule.
- *date_suffix* is the date value automatically supplied by syslogd for sequential data sets to make them unique. The format of this suffix is D*yymmdd*.
- *time_suffix* is the time of day value automatically supplied by syslogd for sequential data sets to make them unique. The format of this suffix is T*hhmmss*.

The following example shows a sequential data set name:

```
USER1.SYSARCH.LOG.D080701.T081342
```

You can repeat this statement multiple times. Each specified statement applies to the rules that follow it, until another instance of this statement is specified. Each statement completely replaces the values from a previous statement. The following sample shows a syslogd configuration file:

```
BeginArchiveParms
  DSNPrefix         USER1.SYSTRACE
  Unit              SYSDA
EndArchiveParms
ArchiveThreshold    80

daemon.debug        /var/syslog/logs/daemon.trace -N DAEMON(+1)
local1.debug        /var/syslog/logs/local1.trace -N LOCAL1(+1)

BeginArchiveParms
  DSNPrefix         USER1.SYSLOG
  Unit              SYSDA
EndArchiveParms

*.*;daemon.none     /var/syslog/logs/syslog.log   -N LOG
```

Given this configuration, the target archive data set names for the configured rules are as follows:
- USER1.SYSTRACE.DAEMON.G*nnnn*V*nn*
- USER1.SYSTRACE.LOCAL1.G*nnnn*V*nn*
- USER1.SYSLOG.LOG.D*yymmdd*.T*hhmmss*

```
►►──BeginArchiveParms──────────────────────────────────────────────►

►──┤ Put Archive Parameters on Separate Lines ├──EndArchiveParms──────────►◄
```

Put Archive Parameters on Separate Lines:

```
├──DSNPrefix prefix──┬───────────┬──┬───────────────┬──┬────────────────┬──────►
                     └─Unit unit─┘  └─Volume volume─┘  └─MgmtClas class─┘

►──┬────────────────┬──┬─────────────┬────────────────────────────────────┤
   └─StorClas class─┘  └─RetPd days──┘
```

DSNPrefix
Specifies the archive data set name prefix value.

Rules:

- The maximum number of characters for the data set prefix depends on the type of data set. For a GDG data set, the maximum length is 35 characters, and for a sequential data set, the maximum length is 28 characters. This length provides space for the other components of the data set name to be supplied. The maximum length of an MVS data set name is 44 characters.
- The data set prefix must conform to the rules for MVS data set names. The maximum length of any component of the data set name is 8 characters. Each component of the name must be composed of alphanumeric or national characters ($ # @). See the DD statement information in the z/OS MVS JCL Reference for more information about data set naming rules.
- The data set prefix can contain system symbolics, for example, &SYSNAME..ARCHIVE. See the coding symbols in JCL in the z/OS MVS JCL Reference for information about using system symbols.

`Unit`
: Specifies the unit information for the dynamic allocation of the target data set. The format of this parameter should conform to the UNIT parameter on the DD JCL statement. This parameter is optional.

`Volume`
: Specifies the volume information for the dynamic allocation of the target data set. The format of this parameter should conform to the VOLUME parameter on the DD JCL statement. This parameter is optional.

`MgmtClas`
: Specifies the management class information for the dynamic allocation of the target data set. The format of this parameter should conform to the MGMTCLAS parameter on the DD JCL statement. This parameter is optional.

`StorClas`
: Specifies the storage class information for the dynamic allocation of the target data set. The format of this parameter should conform to the STORCLAS parameter on the DD JCL statement. This parameter is optional.

`RetPd`
: Specifies the retention period in days for the dynamic allocation of the target sequential data set. Valid values are in the range 0 - 9 999. This parameter is ignored for GDG data sets. The format of this parameter should conform to the RETPD parameter on the DD JCL statement. This parameter is optional.

Syslogd rule configuration statement

This topic describes the syslogd rule configuration statement and associated information.

See “Rule configuration statement” on page 926 for a syntax diagram of the rule configuration statement.

Supported facility names for syslogd

The following facility names are supported and predefined in the syslogd implementation:

user	Message generated by a process (user).
mail	Message generated by mail system.
news	Message generated by news system.
uucp	Message generated by UUCP system.

daemon
This facility name is generally used by server processes. The FTPD server, the RSHD server, the REXECD server, the SNMP agent, and the SNMP subagent use this facility name to log trace messages.

auth/authpriv
Message generated by authorization daemon.

cron Message generated by the clock daemon.

lpr Message generated by the (USS lp command) print client.

local0-7
Names for local use. The z/OS UNIX Telnet server uses the local1 facility name for its log messages.

mark Used for logging MARK messages.

kernel z/OS does not generate any log messages with the kernel facility, and it does not accept log messages from local applications with the kernel facility. However, syslogd on z/OS is capable of receiving log messages over the network from other syslog daemons using the kernel facility. The kernel facility can be used in rules to direct these log messages to specific destinations.

Facilities used by z/OS Communications Server

Table 55 shows the facilities used by z/OS Communications Server.

Table 55. syslogd facilities

Application	syslogd record identifications	Primary syslog facility	Other syslog facility
Application Transparent Transport Layer Security (AT-TLS)	TTLS	daemon	auth
Automated domain name registration (ADNR)	adnr	daemon	None
Communications Server SMTP (CSSMTP)	CSSMTP	mail	None
Defense Manager daemon (DMD)	DMD	local4	None
FTP server	ftpd, ftps	daemon	None
IKE daemon	IKED	local4	None
Network security services (NSS) server	NSSD	local4	None
Network SLAPM2 subagent	NSLAPM2	daemon	None
OMPROUTE	omproute	user	None
OPORTMAP server	oportmap	daemon	None
OREXECD	rexecd	daemon	auth
ORSHD	rshd	daemon	auth
OTELNETD	telnetd	local1	auth
Policy Agent	Pagent	daemon	None

Table 55. syslogd facilities (continued)

Application	syslogd record identifications	Primary syslog facility	Other syslog facility
POPPER	popper	mail	None
PWCHANGE command	pwchange	daemon	None
PWTOKEY command	pwtokey	daemon	None
rpcbind	rpcbind	daemon	None
SENDMAIL	sendmail	mail	None
Simple Network Time Protocol daemon	sntpd	daemon	None
SNMP agent (OSNMPD)	snmpagent	daemon	None
syslogd	syslogd	daemon	None
TCP/IP subagent	M2SubA	daemon	None
TFTP server	tftpd	user	None
TIMED daemon	timed	user	None
TN3270E Telnet subagent	TNSubA	daemon	None
Traffic Regulation Management Daemon (TRMD)	TRMD	daemon (used for IDS logging)	local4 (used for IPSEC logging and defensive filter logging)
Trap Forwarder daemon	trapfwd	daemon	None
z/OS Load Balancing Advisor	lbadv	daemon	None
z/OS Load Balancing Agent	lbagent	daemon	None

Priority codes

When you specify a priority code, all messages with that priority and higher are logged at the specified destination. For example, if you specify a priority code of crit, all messages having alert, panic, emerg, and crit priorities are logged.

The following priority codes are supported. They are shown in priority sequence.

emerg/panic
A panic condition was reported to all processes.

alert A condition that should be corrected immediately.

crit A critical condition.

err(or) An error message.

warn(ing)
A warning message.

notice A condition requiring special handling.

info A general information message.

debug A message useful for debugging programs.

none Do not log any messages for the facility.

* Place holder used to represent all priorities.

Supported destinations for syslogd

The following destinations are supported. File names are case sensitive:

/file A specific path (for example, /tmp/syslogd/auth.log). All log files used by syslogd must be created in the z/OS UNIX file system before syslogd is started unless the **-c** start option is specified. If the **-c** option is specified, the file name can be followed by the -F and -D parameters.

You should use a cron job to signal syslogd at midnight, along with date stamps in the log file directory names, to organize log files by year (%Y), month (%m), and day (%d), only if you do not use the automatic archive function of syslogd. See z/OS Communications Server: IP Configuration Guide for more information about configuring syslogd for automatic archiving. Because both of these methods rely on creating new log files, results could be unpredictable if you try to use both methods together.

The -F parameter specifies the access permissions (modes) for the file if the file must be created dynamically. This parameter has no effect if the file already exists.

Restriction: You cannot specify the -F parameter with the -N or -X parameter.

The -D parameter specifies the access permissions (modes) for the directory part of the file name if the directory (or directories) containing the file must be created dynamically. This parameter has no effect on a directory that already exists.

Restriction: You cannot specify the -D parameter with the -N or -X parameter.

The value following the -F parameter or the -D parameter uses the same octal values as specified for the start options -F and -D. If the -F and -D options are specified on a rule, these values override, for this rule only, the default values specified by the start options. For example, for syslogd to create the file (and directories if needed) for /tmp/syslogd/auth.log, you could specify a rule like the following example:

```
auth.*  /tmp/syslogd/auth.log -F 640 -D 770
```

The permissions in the previous example rule give the owner read/write access to the file and give members of the file's group read-only access. The file’s owner ID is set to the process’s effective user ID (UID), which for syslogd is always UID 0. By default, the owning group ID (GID) is set to that of the parent directory. However, if the FILE.GROUPOWNER.SETGID profile exists in the UNIXPRIV class, the owning GID is determined by the set-GID bit of the parent directory, as follows:

- If the parent’s set-gid bit is on, the owning GID is set to that of the parent directory.
- If the parent’s set-gid bit is off, the owning GID is set to the effective GID of the process.

If the /tmp or the /tmp/syslogd directories do not exist, they are created with access permissions of 770.

You can specify the -N parameter following the file name to specify automatic archival options. The -N parameter is mutually exclusive with the -X parameter.

You should use the -N or -X parameter only if you do not use a cron job to signal syslogd at midnight, along with date stamps in log file directory names, to organize log files by year (%Y), month (%m), and day (%d). See z/OS Communications Server: IP Configuration Guide for more information about offloading log files. Because both of these methods rely on creating new log files, results could be unpredictable if you try to use both methods together.

Results:

- If you specify the -N parameter multiple times on the same rule, the last instance is used.
- If you have multiple rules that use the same destination file, and you specify a mixture of -N and -X parameters on those rules, the parameter you specify on the first such rule is used.

Restriction: You cannot specify the -N parameter with the -F or -D parameter.

The -N parameter specifies a unique qualifier to append to the data set prefix specified on the previous instance of the BeginArchiveParms statement. This prefix forms the base archive data set name. Additional information is appended to the base name to form the complete archive data set name. The format of the additional information depends on the type of data set. You can specify either a GDG or a sequential data set.

The syslogd application requires the correct SAF authorization to create the target data sets that are needed for archival purposes.

For a GDG data set, specify (+1) at the end of the qualifier value. For example, -N TRACE(+1). The GDG specifiers (+0) and (-n) are not valid. The complete archive data set name for a GDG data set is:

prefix.qualifier.gdg_suffix

where:

- *prefix* is the value specified on the BeginArchiveParms statement.
- *qualifier* is the value specified on the -N parameter.
- *gdg_suffix* is the value automatically supplied for GDG data sets to make them unique.

If you use GDG data sets as an archive destination, the GDG BASE must already have been created. Also, be aware of the maximum number of generation data sets to be kept for the GDG. It is possible for syslogd to write more than one archive to the GDG per day, because of the multiple triggers used to perform archives. For example, if you keep five generation data sets, and syslogd performs five archives in one day, you are effectively retaining only a single day's worth of data.

See the information about configuring syslogd for automatic archival in z/OS Communications Server: IP Configuration Guide for sample JCL to create a GDG BASE. See z/OS DFSMS Using Data Sets for more information about GDG data sets.

For a sequential data set do not specify the GDG indicator (+1). The complete archive data set name for a sequential data set is as follows:

prefix.qualifier.date_suffix.time_suffix where:

- *prefix* is the value specified on the BeginArchiveParms statement.
- *qualifier* is the value specified on the -N parameter.
- <date_suffix> is the date value automatically supplied by syslogd for sequential data sets to make them unique. The format of this suffix is: D*yymmdd*.
- *time_suffix* is the time of day value automatically supplied by syslogd for sequential data sets to make them unique. The format of this suffix is T*hhmmss*.

For example, to make a z/OS UNIX file eligible for automatic archival, you could specify the following rule:

```
auth.* /tmp/syslogd/auth.log -N TRACE
```

You can specify the-X parameter following the file name to indicate that the contents of the z/OS UNIX file should be deleted when an archive event occurs. This effectively reinitializes the file without saving the contents. The -X parameter is mutually exclusive with the -N parameter.

Restriction: You cannot specify the -X parameter with the -F or -D parameter.

@host A syslog daemon on another host (for example, @host.domain).

user1,user2,...
A list of users.

/dev/console
The MVS console.

/dev/operlog
The MVS operlog log stream. See the information about system logger applications in z/OS MVS Setting Up a Sysplex.

Requirement: The MVS operlog stream must be active for syslogd to be able to write to it.

SMF record changes

$SMF The log message is stored in SMF record type 109. See the information about type 109 SMF records in z/OS Communications Server: IP Programmer's Guide and Reference for a description of type 109 SMF records. The maximum SMF message is 4096. If the BPX.SMF facility is defined, then the user ID with which syslogd runs must be permitted to BPX.SMF. See SEZAINST(EZARACF) for more information.

- For example, to send all log messages of severity critical or higher from bpxroot or uswmaint to SMF, use the following statement.

```
bpxroot.*.*.crit;uswmaint.*.*.crit   $SMF
```

Usage notes for syslogd

- If you run two instances of syslogd, one for local messages and another for network messages, and you also configure the automatic archival function, do not configure the same UNIX file destinations in the two configuration files. The archival function renames, closes, and reopens the UNIX files. If two instances of syslogd are performing the archival function on the same set of files, results of the archival function are unpredictable. The same is true for the configured archive destination data sets. Be sure to configure unique UNIX file destinations and archive data set names for the two syslogd instances.

- When you specify a priority code, all messages with that priority *and higher* are logged at the specified destination. For example, if you specify a priority code of crit, all messages having alert, panic, emerg, and crit priorities are logged. To send all messages with a priority of crit or higher to a user ID of OPER1, you can enter the following rule in /etc/syslog.conf:

  ```
  *.crit  OPER1
  ```
- You can combine logging rules and destinations in different ways. For example, to send all messages from the facility name daemon into one file and all messages with a priority of crit or higher into another file, enter the following code:

  ```
  daemon.*  /tmp/syslogd/daemon.log
  *.crit    /tmp/syslogd/crit.log
  ```

  Guideline: If a server sends a message to syslogd with a facility name of daemon and a priority code of crit, the message is logged in both the daemon.log and crit.log files. Likewise, if a server sends a message to syslogd with a facility name of daemon and a priority code of alert, the message is logged in both files.
- Syslogd rules may contain a series of conditions. Each condition is separated from the previous condition by a semicolon. For example, if you want to log all messages from facility name local1 or facility daemon into one file, use the following code:

  ```
  local1.*;daemon.* /tmp/syslogd/local1_daemon.log
  ```
- A priority code of none tells syslogd not to select any messages for the specified facility. For example, if you want to log all messages from facility name local1 into one file, all messages from the daemon into another file, and all remaining messages into a third file, use the following code:

  ```
  local1.*                      /tmp/syslogd/local1.log
  daemon.*                      /tmp/syslogd/daemon.log
  *.*;local1.none;daemon.none   /tmp/syslogd/the_rest.log
  ```

  Conditions with a priority of none are called exclusionary conditions. When using syslogd rules with a series of conditions separated by semicolons, all of the individual conditions are evaluated left-to-right for each message. Each matching condition results in either a TRUE (meaning log the message) or a FALSE (meaning don't log the message). Conditions that don't match are ignored. The final result of evaluating each matching condition left-to-right is the result of the last matching condition. Rules that have no matching conditions for a message result in a FALSE. Matching exclude conditions (those with priority of none) result in a FALSE. As an example, consider the difference between the following two rules for a message with facility of daemon and a priority of emerg.

  ```
  daemon.none;*.emerg  /tmp/mylogfile
  *.emerg;daemon.none  /tmp/mylogfile
  ```

  The first rule, first condition, results in FALSE. The first rule, second condition, results in TRUE. Therefore, the message will be logged to the destination for this rule. The second rule, first condition, results in TRUE. The second rule, second condition, results in FALSE. The message will not be logged for this rule.

  The order of conditions within the filter is significant.

  Guideline: You should set up the /tmp/syslogd directory as a separate z/OS UNIX file system. Unless managed properly, the syslogd daemon can fill up the /tmp hfs, which can impact other applications that might require temporary space in the /tmp directory.

- You can define logging conditions that contain a userid and jobname along with the facility and priority. The user ID value, the job name, or both can be specified as an asterisk (*), which matches any user ID or any job name.

  Restrictions:

  – Only messages that are issued by a program running under the specified user ID or job name and that also match the facility and priority are logged.

  – The user ID and job name filter is used only for messages that originate on the same system where syslogd is running. The filter does not apply for messages received from the IP network.

  For example, if you want to log all messages from programs running under userid USER1 (with any jobname, facility or priority) to one file and log all messages from any userid with jobname JOB1 with facility of daemon and any priority to another file, use the following code:

```
user1.*.*.*                      /tmp/syslogd/user1.log
*.job1.daemon.*                  /tmp/syslogd/job1.daemon.log
```

- You can define logging conditions that contain an IP address or host name along with the facility and priority. If you use a host name, it must be able to be resolved to an IP address.

  Restriction: Only messages received over the IP network use a filter containing an IP address or host name.The IP address or host name filter is not used for local messages received over the syslog AF_UNIX socket. For example, if you want to log all messages from host1.*xyz*.com to one file and all messages from 192.168.0.1 with facility daemon and priority info or higher to another file, use the following code:

```
(host1.xyz.com).*.*                 /tmp/syslogd/host1.log
(192.168.0.1).daemon.info           /tmp/syslogd/host2.log
```

- It is possible to create rules that contain an IP address (or host name) in one condition along with *userid.jobname.facility.priority* in another condition.

  Rule: Conditions must be separated by semicolons.

- If using IP addresses in conditions, the address can be followed by an optional forward slash and a number representing the number of significant bits of the address. This is called the prefix length. The prefix length provides a means to indicate that a condition applies to all IP addresses that have the bit pattern for the specified number of bits. For example, the following rule matches all messages received from IP addresses 192.168.0.0 - 192.168.0.255 that also have a facility of daemon and a priority of info or higher:

```
(192.168.0.1/24).*.*                 /tmp/syslogd/host1.log
```

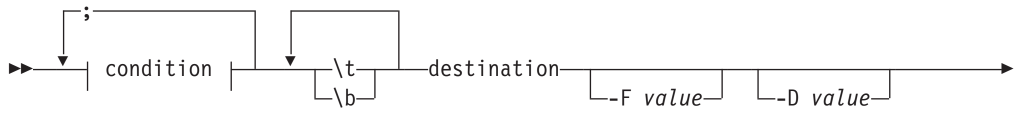
- IPv6 addresses or host names that resolve to IPv6 addresses can be used in the rule conditions or as destinations (if forwarding to another host).

  Restriction: Do not use IPv4-mapped IPv6 addresses or IPv6 addresses with the reserved prefix ::/96.

Syntax

Each rule statement of the configuration file has the following syntax:

Rule configuration statement



```
►──┬─────────────┬──┬────┬──────────────────────────────────────────────►◄
   └─-N qualifier─┘  └─-X─┘
```

condition (option 1):

```
   ┌─;──────────────────────────────────────────┐
   ▼                                              │
├──┬────────────────────┬─facility─.─priority─┴──────────────────────────┤
   └─userid . jobname.─┘
```

condition (option 2):

```
   ┌─;─────────────────────────────────────┐
   ▼                                         │
├──┬──────────────┬─facility─.─priority─┴──────────────────────────────────┤
   └─(hostspec).─┘
```

Parameters

hostspec

- An IPv4 address, for example, 192.168.0.1
- An IPv4 address with a subnet length, for example, 192.168.0.1/24
- An IPv6 address, for example, FEC0::9:42:105:19
- An IPv6 address with a prefix length, for example, FEC0::9:42:105:19/64
- A host name that resolves to an IPv4 or IPv6 address. The host name can be specified with or without the DNS suffix. If specified without the suffix, the suffix is assumed to be one of the suffixes defined to resolver for the host where syslogd is running.

facility
See "Supported facility names for syslogd" on page 919 for details.

priority
See "Priority codes" on page 921 for details.

destination
See "Supported destinations for syslogd" on page 922 for details.

-F When the destination is a UNIX file, this parameter specifies the access permissions (modes) for the file if the file must be created dynamically. See the description of the /file destination in "Supported destinations for syslogd" on page 922 for details.

-D When the destination is a UNIX file, this parameter specifies the access permissions (modes) for the directory part of the file name if the directory (or directories) containing the file must be created dynamically. See the description of the /file destination in "Supported destinations for syslogd" on page 922 for details.

-N When the destination is a UNIX file, this parameter specifies a unique qualifier to append to the archive data set prefix specified on the previous instance of the BeginArchiveParms statement. The existing UNIX file is left empty and ready to receive more messages. The original contents of the file are moved into the specified archive dataset. See the description of the /file destination in "Supported destinations for syslogd" on page 922 for details.

Result: The destination log file will be deleted as a part of the archival process, and a new one created at the next use. Ensure that the desired permission bits and file ownership attributes are set as described in Setting permissions for log files and directories in z/OS Communications Server: IP Configuration Guide.

-X When the destination is a UNIX file, this parameter indicates that the contents of the z/OS UNIX file should be deleted when an archive event occurs. The existing UNIX file is left empty and ready to receive more messages. The original contents of the file are lost. Even though -X is not archiving a file, it is still governed by archive events. See the description of the /file destination in "Supported destinations for syslogd" on page 922 for details.

Result: The destination log file will be deleted as a part of the archival process, and a new one created at the next use. Ensure that the desired permission bits and file ownership attributes are set as described in Setting permissions for log files and directories in z/OS Communications Server: IP Configuration Guide.

Restrictions:
- The -F, -D, -N, and -X parameters are only valid when the destination is a z/OS UNIX file system file.
- You cannot specify scope information as part of an IP address or a host name.
- You cannot specify the -F or -D parameter with the -N or -X parameter.
- You cannot specify the -N parameter with the -X parameter.

The \t parameter in the syntax diagram is a tab character; the \b parameter is a blank space.

Syslogd browser tool

The two steps to enable the syslogd browser ISPF interface are as follows:
1. Provide ISPF library access. You must provide access to the z/OS Communications Server ISPF libraries. You can do this by modifying the TSO logon procedures or by running a CLIST.
2. Add the syslogd browser to the ISPF Primary Option menu or to an ISPF options menu of your choice. To be able to select the syslogd browser interface from your ISPF Primary Option menu, you need to update the menu and processing sections of the ISR@PRIM panel.

Requirement: You must be able to scroll forward and backward in the ISPF interface to access specific information. Be sure that your keyboard has specific keys for Page Up and Page Down or that you have set PF keys for these functions using option 0.3 on the ISPF Primary Option menu. UP or FORWARD works for scrolling forward. DOWN, BACK, or BACKWARD works for scrolling back.

Providing library access

You must provide access to the z/OS Communications Server ISPF libraries. You can do this by performing either of the following actions:
- Adding DD statements to your TSO logon procedure
- Allocating the libraries with a REXX exec

The following ISPF libraries are required for using the syslogd browser ISPF interface:

- *hlq*.SEZAPENU (ISPF panel library; member names all start with EZASY)
- *hlq*.SEZAMENU (ISPF message library; member names all start with EZASY)
- *hlq*.SEZAEXEC (REXX program library)

Using the TSO logon procedure

One method of providing access to the z/OS Communications Server ISPF libraries is to add them to the TSO logon procedure.

Add the following DD statements to your TSO logon procedure, and replace *hlq* with your installation's high level qualifier for z/OS Communications Server libraries:

```
//ISPPLIB   DD DSN=hlq.SEZAPENU,DISP=SHR
//ISPMLIB   DD DSN=hlq.SEZAMENU,DISP=SHR
```

and

```
//SYSEXEC DD DSN=hlq.SEZAEXEC,DISP=SHR
```

or

```
//SYSPROC DD DSN=hlq.SEZAEXEC,DISP=SHR
```

Using a CLIST

Another method of providing access to the z/OS Communications Server ISPF libraries is to run a CLIST or a REXX program to allocate the z/OS Communications Server ISPF libraries. Copy *hlq*.SEZAEXEC(EZABROWS) into your system CLIST or REXX library and make changes as indicated in the comments of that member.

Adding the syslogd browser to the ISPF primary option menu

ISR@PRIM is the default ISPF Primary Option menu panel and a member in the ISPPLIB library. If you want all of your users to have access to the syslogd browser from the ISPF primary option menu, you must update the ISR@PRIM member in the following two places:

- In the menu section (Part 1 of ISR@PRIM) to have an option for the syslogd browser appear on the ISPF Primary Option menu. See the example in this topic.
- In the processing section (Part 2 of ISR@PRIM) so that the selection invoke the syslogd browser ISPF interface. You can optionally have the selection execute the initialization CLIST before invoking the syslogd browser. See the examples in Figure 31 on page 930 and Figure 32 on page 930.

After you update ISR@PRIM, the option that you added for the syslogd browser appears on the ISPF Primary Option menu after the next ISPF logon.

Figure 31 on page 930 shows the menu section of the ISPF Primary Option menu for ISR@PRIM.

```
)AREA SAREA39
.0 .Settings     .Terminal and user parameters         .
.1 .View         .Display source data or listings      .
.2 .Edit         .Create or change source data         .
.3 .Utilities    .Perform utility functions            .
.4 .Foreground   .Interactive language processing      .
.5 .Batch        .Submit job for language processing   .
.6 .Command      .Enter TSO or Workstation commands    .
.7 .Dialog Test  .Perform dialog testing               .
.9 .IBM Products .IBM program development products     .
.10.SCLM         .SW Configuration Library Manager     .
.11.Workplace    .ISPF Object/Action Workplace         .
.12.z/OS System  .z/OS system programmer applications  .
.13.z/OS User    .z/OS user applications               .
.14.Syslogd      .z/OS CS Syslogd browser              .   <=======
```

Figure 31. Menu section of the ISPF primary option menu for ISR@PRIM

Figure 32 shows the processing section of the ISPF Primary Option menu for ISR@PRIM.

```
 &ZSEL = TRANS (TRUNC (&ZCMD,'.')
  0,'PGM(ISPISM) SCRNAME(SETTINGS)'
  1,'PGM(ISRBRO) PARM(ISRBRO01) SCRNAME(VIEW)'
  2,'PGM(ISREDIT) PARM(P,ISREDM01) SCRNAME(EDIT)'
  3,'PANEL(ISRUTIL) SCRNAME(UTIL)'
  4,'PANEL(ISRFPA) SCRNAME(FOREGRND)'
  5,'PGM(ISRJB1) PARM(ISRJPA) SCRNAME(BATCH) NOCHECK'
  6,'PGM(ISRPTC) SCRNAME(CMD)'
  7,'PGM(ISPYXDR) PARM(&ZTAPPLID) SCRNAME(DTEST) NOCHECK'
  9,'PANEL(ISRDIIS) ADDPOP'
 10,'PGM(ISRSCLM) SCRNAME(SCLM) NOCHECK'
 11,'PGM(ISRUDA) PARM(ISRWORK) SCRNAME(WORK)'
 12,'PANEL(ISR@390S) SCRNAME(OS390S)'
 13,'PANEL(ISR@390U) SCRNAME(OS390U)'
 14,'CMD(EZASYRGO) NEWPOOL PASSLIB NEWAPPL(EZAS)" '      <======
  X,EXIT
```

Figure 32. Processing section of the ISPF Primary Option menu for ISR@PRIM

Chapter 21. Policy Agent and policy applications

For related information about Policy Agent and the policy applications, see the policy based networking information in z/OS Communications Server: IP Configuration Guide.

Also, for more information about policy schema definition files, see Chapter 23, “Intrusion detection services policy,” on page 1225.

Policy configuration files

This topic contains information about the policy configuration files, including overviews and syntax rules.

Policy Agent configuration files overview

When the Policy Agent is started, the main policy configuration file is used. Use this initial configuration file to point to other policy files that contain specific policies for other corresponding TCP/IP images. Policy files can be stored locally or on a remote system.

There are multiple types of configuration files. The role of the Policy Agent determines which files are used and for what purpose. Policy Agent can act in the role of a policy server, providing centralized policy services for a set of policy clients. It can also act as a policy client, retrieving remote policies from the policy server. The policy client installs these remote policies in the corresponding TCP/IP image. In either case (policy client or policy server), local policies can also be stored in local configuration files. The following configuration files are used on the policy client or policy server to configure operational characteristics or to define local policies:

- Main configuration file (determined using a standard search order); can refer to policy common configuration files
- Common IPSec configuration file (specified on the CommonIpSecConfig statement)
- Common Application Transparent Transport Layer Security (AT-TLS) configuration file (specified on the CommonTTLSConfig statement)
- Common IDS configuration file (specified on the CommonIDSConfig statement)
- Common Routing configuration file (specified on the CommonRoutingConfig statement)
- Image configuration files (specified on the TcpImage or PEPInstance statement); can refer to policy specific image configuration files
- Image IPSec configuration files (specified on the IpSecConfig statement)
- Image AT-TLS configuration files (specified on the TTLSConfig statement)
- Image IDS configuration files (specified on the IDSConfig statement)
- Image QoS configuration files (specified on the QOSConfig statement)
- Image Routing configuration files (specified on the RoutingConfig statement)

On the policy server, the following configuration files are used to define policies for policy clients:

- Policy client common configuration files (specified on the DynamicConfigPolicyLoad statement)

- Policy client image configuration files (specified on the DynamicConfigPolicyLoad statement

Tips:

- If the TcpImage or PEPInstance statement does not specify an image configuration file, then the main configuration file is also the image configuration file for that TCP/IP image.
- If the QOSConfig statement is not specified, then QoS policies are defined in the image configuration file, not in a separate image QoS configuration file.

Policy Agent configuration statements overview

The following statements configure basic operational parameters for the Policy Agent, and you can specify them only in the main configuration file:

- AutoMonitorApps
- AutoMonitorParms
- ClientConnection (on the policy server)
- Codepage
- DynamicConfigPolicyLoad (on the policy server)
- LogLevel
- ServerConnection (on the policy client)
- ServicesConnection
- TcpImage or PEPInstance

Use the following statement to configure a policy client to retrieve remote policies from a policy server. Specify this statement in the image configuration file, on a per-stack basis:

- PolicyServer

Use the following statements to configure optional files for some policy types (both common and image-specific files) to obtain local policies:

- CommonIDSConfig
- CommonIPSecConfig
- CommonRoutingConfig
- CommonTTLSConfig
- IdsConfig
- IPSecConfig
- QOSConfig
- RoutingConfig
- TTLSConfig

The ReadFromDirectory statement optionally configures the Policy Agent as an LDAP client, and you can specify them in the image configuration files, on a per-stack basis.

The following statements configure functional parameters for the Policy Agent and you can specify them in the image QoS configuration files, on a per-stack basis:

- PolicyPerfMonitorForSDR
- PolicyPerformanceCollection
- SetSubnetPrioTosMask

Table 65 on page 945 shows statements that define policies; you can specify these statements in the image configuration files for each policy type, on a per-stack basis.

General syntax rules for Policy Agent

The following list shows the general configuration rules. Unless otherwise noted, these rules apply to both the configuration file and the Lightweight Directory Access Protocol (LDAP) server:

- Specify Policy Agent configuration files using code page IBM-1047 for EBCDIC, unless the Codepage statement is configured.
- Only one attribute and its values can be specified per line.
- Text beyond the specified attribute and value is ignored.
- Text beginning with the # character is a comment and is ignored, unless documented otherwise.
- Comments beginning with the # character in an LDAP server ldif configuration file might only be recognized as comments at the beginning of the file; therefore do not specify such comments elsewhere in the file, as they are interpreted as part of an attribute or attribute value.
- For most range specifications, the ranges can be delimited by a colon (:), a dash (-), or a blank ( ), but these delimiters cannot be mixed within a single range specification. IP address ranges cannot use the colon or blank delimiter, unless stated otherwise.
- See z/OS Communications Server: IPv6 Network and Application Design Guide for information about types of policies that support IPv6.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.
- IPv4-mapped IPv6 addresses and IPv6 addresses with the reserved prefix ::/96 are valid only for IP filter rules and for the Identity parameter on local and remote security end points.
- The maximum decimal value for numeric values is 4294967295, unless otherwise noted.
- Policy rule and action names are limited to 32 characters. If QoS and IDS LDAP statement names longer than 32 characters are specified, they are silently truncated. All other statements longer than 32 characters cause an error message to be written to the log.
- If a configuration file or LDAP configuration contains duplicate statement or object names, Policy Agent keeps the first or the last statement or object, as follows. The following situations are considered warnings, not errors.
  - For IDS (LDAP) and QoS, Policy Agent keeps the first entry.
  - For IDS (configuration file), IPSec, Routing, and AT-TLS, Policy Agent keeps the last entry.
- If a QoS or IDS statement or object is defined with the same name in both a configuration file and LDAP, Policy Agent keeps the first such statement or object that it reads. This is typically the statement or object in the configuration file, but as a result of timing constraints, it could also be the statement in LDAP. The last duplicate statement or object is discarded; this is considered an error.
- Specify most attributes for configuration file statements only once per statement (exceptions are noted where appropriate). If you specify multiple attributes, no error or warning messages are written to the log, and the last instance of the attribute is used.
- Attributes for policies defined on an LDAP server can be single- or multi-valued (meaning a single value or multiple values are allowed for that attribute). The

Policy Agent detects multiple values for attributes that are defined as single valued, and treats the policy object as in error.

- The policy version is specified by the configuration file statement name, as follows:
  - ServicePolicyRules and ServiceCategories statements specify version 1 policies.
  - PolicyRule and PolicyAction statements specify version 2 policies.

    Result: The policy version of LDAP-defined objects is determined by the LDAP_SchemaVersion parameter on the ReadFromDirectory statement.

  For more information about policy version definitions, see z/OS Communications Server: IP Configuration Guide. For more information about policy version differences, see z/OS Communications Server: IP Diagnosis Guide.
- Some configuration statements use an inline statement syntax. When a given statement is specified inline within another statement, only the inline statement name is shown in the syntax diagrams. However, the entire statement being inlined must be specified, including its own set of start and end braces ({}) and all parameters.

  Tip: The **name** parameter on the statement name might or might not be optional, depending on the specific statement. In the following example, the IpFilterRule statement is included inline within the IpFilterGroup statement. A name is required on the IpFilterRule statement, for example, Rule1All-Permit, as follows:

  ```
  IpFilterGroup ZoneAll
  {
    IpFilterRule Rule1All-Permit
    {
    IpSourceAddr All
    IpDestAddr All
    IpServiceGroupRef Resolver
    IpServiceRef PathMtuDiscovery
    IpServiceGroupRef Ping-Outbound-Only
    IpGenericFilterActionRef permit
    }
  }
  ```
- For named inline statements where the name is optional, a nonpersistent system name is created using the named portion of the statement name with a unique identifier. This prevents reuse of the named inline statement as a reference name.
- Errors detected in a policy rule or action result in that policy object being discarded.
- For IPSec, Routing, or AT-TLS policies, any errors detected during parsing results in no new policies being installed. For all other policy types, only the policy objects that contain errors are discarded.
- If a rule refers to an action that does not exist (or has been discarded due to an error) then the rule is also discarded.
- If a Routing action refers to a route table that does not exist (or has been discarded as the result of an error), the action is also discarded.
- Some statements, parameters, parameter values, rules, or restrictions apply only to certain release levels. See the Policy-based networking information in z/OS Communications Server: IP Configuration Guide for more details about mixed release levels when using policy clients with a policy server. The following tables list the definitions that are supported for each release level.

Table 56 lists statements, parameters, and parameter values that are no longer supported:

Table 56. Statements, parameters, and parameter values that are no longer supported

Statement	Parameter	Parameter value	Description/Notes	Last release supported
PolicyAction	PolicyScope	TR	TR indicates that the scope is Traffic Regulation.	z/OS V1R9
PolicyAction	• TypeActions • TotalConnections • Percentage • TimeInterval • LoggingLevel			z/OS V1R9
IpFilterPolicy	RFC4301Compliance		This parameter is deprecated. RFC 4301 compliance is no longer optional as of V1R12.	z/OS V1R11

Table 57. Valid statements, parameters, and parameter values for z/OS V2R1 and later releases

Statement	Parameter	Parameter value	Description of change
RouteTable	DynamicRoutingParms	*gateway_addr*	Value can be an IPv4 address, an IPv6 address, the keyword IPV4, or the keyword IPV6.
RouteTable	Route	*ipaddress*	Value can be an IPv4 address, an IPv6 address, the keyword DEFAULT, or the keyword DEFAULT6.
RouteTable	Route	*gateway_addr*	Value can be an IPv4 address or an IPv6 address.
RouteTable	Multipath6		
RouteTable	DynamicXCFRoutes6		
RouteTable	IgnorePathMtuUpdate6		
RoutingRule	IpSourceAddr	*ipaddress*	Value can be an IPv4 address, an IPv6 address, or the keyword All. If a source address is not specified, the default value is All.
RoutingRule	• IpSourceAddrRef • IpSourceAddrSetRef • IpSourceAddrGroupRef		An IPv4 addresses, an IPv6 addresses, or both, can be referenced. If a source address is not specified, the default value is All.

Table 57. Valid statements, parameters, and parameter values for z/OS V2R1 and later releases (continued)

Statement	Parameter	Parameter value	Description of change
RoutingRule	IpDestAddr	*ipaddress*	Value can be an IPv4 address, an IPv6 address, or the keyword All. If a destination address is not specified, the default value is All.
RoutingRule	• IpDestAddrRef • IpDestAddrSetRef • IpDestAddrGroupRef		An IPv4 addresses, an IPv6 addresses, or both, can be referenced. If a destination address is not specified, the default value is All.
TTLSCipherParms	• V3CipherSuites • V3CipherSuites4Char	See Table 78 for list of new cipher names and 2 or 4 hexadecimal character values.	• V3CipherSuites – New 2-hexadecimal character values – New cipher names defined • V3CipherSuites4Char – New 4-hexadecimal character values
TTLSConnection Action	• TTLSSignatureParms • TTLSSignatureParmsRef		
TTLSConnection AdvancedParms	TLSv1.2	• Off • On	
TTLSEnvironment Action	• SuiteBProfile • TTLSSignatureParms • TTLSSignatureParmsRef	• SuiteBProfile • Off • 128 • 192 • All	
TTLSEnvironment AdvancedParms	• TLSv1.2 • Renegotiation • RenegotiationIndicator • RenegotiationCertCheck	• TTLSv1.2 • Off • On • Renegotiation • Default • Disabled • All • Abbreviated • RenegotiationIndicator • Optional • Client • Server • Both • RenegotiationCertCheck • Off • On	

Table 57. Valid statements, parameters, and parameter values for z/OS V2R1 and later releases (continued)

Statement	Parameter	Parameter value	Description of change
TTLSSignatureParms	• ClientECurves • SignaturePairs		**ClientECurves** Specifies the list of elliptic curves that are supported by the client, in order of preference for use. The elliptical curve specifications are used by the client to tell the server which elliptical curves can be used when using cipher suites that use elliptical curve cryptography for the TLSv1.0 protocol or later. **SignaturePairs** Specifies the TLS version 1.2 signature algorithm pairs that are supported for the server certificate. These pairs are sent by the client when proposing use of the TLSv1.2 protocol to indicate to the server which signature/hash algorithm pairs might be used in digital signatures of the server certificate. SignaturePairs is meaningful only when performing a handshake with a Server that supports the TLSv1.2 protocol and will be ignored by any Server that only supports TLSv1.1 protocol or earlier.

Table 58. Valid statements, parameters, and parameter values for z/OS V1R13 and later releases

Statement	Parameter	Parameter value	Description of change
IDSAction	ActionType Attack	• ResetConn • NoResetconn	

Table 58. Valid statements, parameters, and parameter values for z/OS V1R13 and later releases (continued)

Statement	Parameter	Parameter value	Description of change
IDSAttackCondition	AttackType	• DATA_HIDING • OUTBOUND_RAW_IPV6 • RESTRICTED_IPV6_DST_OPTIONS • RESTRICTED_IPV6_HOP_OPTIONS • RESTRICTED_IPV6_NEXT_HDR • TCP_QUEUE_SIZE • GLOBAL_TCP_STALL • EE_MALFORMED_PACKET • EE_LDLC_CHECK • EE_PORT_CHECK • EE_XID_FLOOD	
IDSAttackCondition	• OptionPadChk • IcmpEmbedPktChk • RestrictedIPv6OptionRange • RestrictedIPv6OptionRangeRef • RestrictedIPv6OptionGroupRef • IPv6NextHdrRange • IPv6NextHdrRangeRef • IPv6NextHdrGroupRef • TcpQueueSize • IDSExclusion • IDSExclusionRef • EEXIDTimeout		
IDSExclusion			
IDSScanEvent Condition	Protocol	• Icmpv6 • 58	
IDSScanEvent Condition	LocalHostAddr	• *ipaddress* • All	Value can be an IPv4 address or an IPv6 address. All includes both IPv4 and IPv6 addresses.
IDSScanExclusion	ExcludedAddrPort	*ipaddress*	Value can be an IPv4 address or an IPv6 address.
IDSTRCondition	LocalHostAddr	• *ipaddress* • All	Value can be an IPv4 address or an IPv6 address. All includes both IPv4 and IPv6 addresses.
Ipv6NextHdrGroup			
Ipv6NextHdrRange			

Table 58. Valid statements, parameters, and parameter values for z/OS V1R13 and later releases (continued)

Statement	Parameter	Parameter value	Description of change
IpAddr	Addr	*ipaddress*	Value can be an IPv4 address or an IPv6 address
IpAddrSet	Prefix Range	*ipaddress*	Value can be an IPv4 address or an IPv6 address

Table 59. Valid statements, parameters, and parameter values for z/OS V1R12 and later releases

Statement	Parameter	Parameter value	Description of change
IpDataOffer	HowToEncap		This is no longer a required parameter. The default is Tunnel.
	HowToEncrypt	• AES • AES_CBC KeyLength 128 • AES_CBC KeyLength 256 • AES_GCM_16 KeyLength 128 • AES_GCM_16 KeyLength 256	AES is deprecated and treated as a synonym for AES_CBC KeyLength 128.
	HowToAuth	• Null • AES128_XCBC_96 • AES_GMAC_128 • AES_GMAC_256 • HMAC_SHA • HMAC_SHA1 • HMAC_SHA2_256_128 • HMAC_SHA2_384_192 • HMAC_SHA2_512_256	• Null is allowed only in combination with HowToEncrypt AES_GCM_16. • AES_GMAC_128 and AES_GMAC_256 are allowed only in combination with HowToEncrypt DoNot. • HMAC_SHA is deprecated and treated as a synonym for HMAC_SHA1.
IpDynVpnAction	HowToEncapIKEv2		
	• InitiateWithPFS • AcceptablePFS	• Group19 • Group20 • Group21 • Group24	
IpFilterPolicy	FIPS140		
IpLocalStartAction	ICMPCodeGranularity		
	ICMPTypeGranularity		
	ICMPv6CodeGranularity		
	ICMPv6TypeGranularity		
	MIPv6TypeGranularity		

Table 59. Valid statements, parameters, and parameter values for z/OS V1R12 and later releases (continued)

Statement	Parameter	Parameter value	Description of change
IpManVpnAction	AuthInboundSa		New values are allowed for the key length, for the new algorithms added to the HowToAuth parameter.
	AuthOutboundSa		New values are allowed for the key length, for the new algorithms added to the HowToAuth parameter.
	EncryptInboundSa		New values are allowed for the key length, for the new algorithms added to the HowToEncrypt parameter.
	EncryptOutboundSa		New values are allowed for the key length, for the new algorithms added to the HowToEncrypt parameter.
	HowToAuth	• AES128_XCBC_96 • HMAC_SHA • HMAC_SHA1 • HMAC_SHA2_256_128 • HMAC_SHA2_384_192 • HMAC_SHA2_512_256	HMAC_SHA is deprecated and treated as a synonym for HMAC_SHA1.
	HowToEncrypt	• AES • AES_CBC KeyLength 128 • AES_CBC KeyLength 256	AES is deprecated and treated as a synonym for AES_CBC KeyLength 128.
KeyExchangeAction	BypassIpValidation		
	CertificateURLLookupPreference		
	HowToAuthMe		
	HowToInitiate	IKEv2	The default for this parameter is now obtained from the HowToInitiate parameter on the KeyExchangePolicy statement.
	HowToRespond		Deprecated and treated as a synonym for HowToRespondIKEv1.
	HowToRespondIKEv1		This is introduced as a more accurate synonym for HowToRespond, which is now deprecated.
	ReauthInterval		
	RevocationChecking		

Table 59. Valid statements, parameters, and parameter values for z/OS V1R12 and later releases (continued)

Statement	Parameter	Parameter value	Description of change
KeyExchangeOffer	DHGroup	• Group19 • Group20 • Group21 • Group24	
	HowToEncrypt	• AES • AES_CBC KeyLength 128 • AES_CBC KeyLength 256	AES is deprecated and treated as a synonym for AES_CBC KeyLength 128.
	HowToAuthMsgs	• SHA2_256 • SHA2_384 • SHA2_512	
	HowToVerifyMsgs		
	PseudoRandomFunction		
KeyExchangePolicy	BypassIpValidation		
	CertificateURLLookupPreference		
	HowToInitiate		
	LivenessInterval		
	RevocationChecking		
LocalSecurityEndpoint	Identity	KeyId	
RemoteIdentity	Identity	KeyId	
RemoteSecurityEndpoint	Identity	KeyId	

Table 60. Valid statements, parameters, and parameter values for z/OS V1R10 and later releases

Statement	Parameter	Parameter value	Description of change
IpDynVpnAction	• PassthroughDF • PassthroughDSCP		
IPFilterPolicy	• ImplicitDiscardAction • RFC4301Compliance		
IPFilterRule	• RemoteIdentity • RemoteIdentityRef		
IpGenericFilterAction	DiscardAction		
IpManVpnAction	• PassthroughDF • PassthroughDSCP		
IpManVpnAction	• LocalSecurityEndpointAddr • RemoteSecurityEndpointAdd	• Any • Any4	
IpService	Protocol	• MIPv6 • Opaque	
IpService	FragmentsOnly		

Table 60. Valid statements, parameters, and parameter values for z/OS V1R10 and later releases (continued)

Statement	Parameter	Parameter value	Description of change
IpService	Type Code		A range of values is allowed when the Protocol parameter value is Icmp or Icmpv6.
IpService	Protocol	IPv6Frag	IPv6Frag is not valid. This IPv6Frag value does not match any traffic.
KeyExchangeAction	• FilterByIdentity • ConstrainSource • ConstrainSourceRef • ConstrainSourceSetRef • ConstrainSourceGroupRef • ConstrainDest • ConstrainDestRef • ConstrainDestSetRef • ConstrainDestGroupRef		
LocalSecurityEndpoint	Location	• ipaddress/ prefixLength • ipaddress-ipaddress	
LocalSecurityEndpoint	• LocationSetRef • LocationGroupRef		
RemoteIdentity			
RemoteSecurityEndpoint	• LocationGroupRef • RemoteIdentityRef		

Table 61 lists statements, parameters, and parameter values that contain rules or restrictions that differ for z/OS V1R12 and later releases, as compared to earlier releases.

Table 61. Valid rules and restrictions for V1R12 and later releases

Statement	Parameter	Parameter value	Description of change
IpService	• Type • Code		The rule about certain Type and Code values not being allowed in combination with the IpDynVpnAction statement is removed.

Table 62 on page 943 lists statements, parameters, and parameter values that contain rules or restrictions that differ for z/OS V1R10 and later releases, as compared to earlier releases.

Table 62. Valid rules and restrictions for V1R10 and later releases

Statement	Parameter	Parameter value	Description of change
IpManVpnAction	• LocalSecurityEndpointAddr • RemoteSecurityEndpointAdd	*address*	The IPv6 and IPv4 unspecified addresses are not allowed.
IpManVpnAction	• AuthInboundSa • EncryptInboundSa	*spi*	In prior releases, IpManVpnAction objects were required to have unique inbound AH or ESP *spi* values. *spi* values no longer need to be unique if the LocalSecurityEndpointAddr specification differs from that of other IpManVpnAction objects that share the same AH or ESP *spi* value.
IpService	• SourcePortRange • DestinationPortRange • Type Code		For V1R12 and later releases, or if RFC4301Compliance Yes is specified on the IpFilterPolicy statement, the Routing specification Routed or Either must have one of the following configurations: • A SourcePortRange and DestinationPortRange specification configured to 0 (if applicable) • A Type and Code specification configured to Any (if applicable)

Policy Agent general configuration file statements

Table 63 and Table 64 on page 944 list the Policy Agent general configuration file statements, including the purpose of each statement.

Table 63. Policy Agent main configuration file statements

Statement	Purpose	See
AutoMonitorApps	Specifies applications to be monitored and automatically started or restarted by Policy Agent.	"AutoMonitorApps statement" on page 950
AutoMonitorParms	Specifies global parameters that control how Policy Agent monitors and starts or restarts applications.	"AutoMonitorParms statement" on page 954
ClientConnection	Configures the Policy Agent as a policy server, listening on the specified port for remote connections.	"ClientConnection statement" on page 955
Codepage	Specifies the EBCDIC code page to be used when reading configuration files and policy definition files.	"Codepage statement" on page 956
CommonIDSConfig	Specifies the path of an IDS policy file that contains common IDS policy statements.	"CommonIDSConfig statement" on page 957

Table 63. Policy Agent main configuration file statements (continued)

Statement	Purpose	See
CommonIPSecConfig	Specifies the path of an IPSec policy file that contains common IPSec policy statements.	"CommonIPSecConfig statement" on page 958
CommonRoutingConfig	Specifies the path of a Routing policy file that contains common Routing policy statements.	"CommonRoutingConfig statement" on page 959
CommonTTLSConfig	Specifies the path of an AT-TLS policy file that contains common AT-TLS policy statements.	"CommonTTLSConfig statement" on page 960
DynamicConfigPolicyLoad	Specifies the configuration file names to use on the policy server for policy client policies.	"DynamicConfigPolicyLoad statement" on page 961
LogLevel	Specifies level of tracing.	"LogLevel statement" on page 970
ServerConnection	Specifies the connection information used by a policy client to connect to the policy server. This statement includes security information and the location of the policy server.	"ServerConnection statement" on page 988
ServicesConnection	Specifies the listening port, listening TCP/IP image, and security level for connections to this Policy Agent.	"ServicesConnection statement" on page 993
TcpImage and PEPInstance	Defines a TCP/IP image and its associated configurations.	"TcpImage and PEPInstance statement" on page 999

Table 64. Policy Agent image configuration file statements

Statement	Purpose	File	See
IDSConfig	Specifies the path of an IDS policy file that contains stack-specific IDS policy statements. This statement is required to read an IDS configuration file for a given stack.	Image	"IDSConfig statement" on page 967
IPSecConfig	Specifies the path of an IPSec policy file that contains stack-specific IPSec policy statements. This statement is required to define IPSec policy for a given stack.	Image	"IPSecConfig statement" on page 969
PolicyPerfMonitorForSDR	Enables or disables the policy performance monitor function.	QoS image	"PolicyPerfMonitorForSDR statement" on page 971
PolicyPerformanceCollection	Enables or disables the policy performance collection function.	QoS image	"PolicyPerformanceCollection statement" on page 974

Table 64. Policy Agent image configuration file statements (continued)

Statement	Purpose	File	See
PolicyServer	Configures the Policy Agent as a policy client, and specifies what types of policies to retrieve from the policy server. This statement also specifies security and processing information that is passed to the policy server.	Image	"PolicyServer statement" on page 977
QOSConfig	Specifies the path of a QoS policy file that contains stack-specific QoS policy statements.	Image	"QOSConfig statement" on page 981
ReadFromDirectory	Initializes Policy Agent as an LDAP client.	Image	"ReadFromDirectory statement" on page 981
RoutingConfig	Specifies the path of a Routing policy file that contains stack-specific Routing policy statements. This statement is required to read a Routing configuration file for a given stack.	Image	"RoutingConfig statement" on page 987
SetSubnetPrioTosMask	Defines IPv4 ToS byte or IPv6 Traffic Class to device and virtual LAN (VLAN) user priority mapping.	QoS image	"SetSubnetPrioTosMask statement" on page 996
TTLSConfig	Specifies the path of an AT-TLS policy file that contains stack-specific AT-TLS policy statements. This statement is required to define AT-TLS policy for a given stack.	Image	"TTLSConfig statement" on page 1001

Table 65 lists the configuration file statements that define policies, and the purpose and policy type of each.

Table 65. Policy Agent configuration file policy statements

Statement	Purpose	Type	See
IDSAction	Defines IDS action.	IDS	"IDSAction statement" on page 1043
IDSAttackCondition	Defines IDS rule attack condition.	IDS	"IDSAttackCondition statement" on page 1046
IDSExclusion	Defines IDS rule exclusion	IDS	"IDSExclusion statement" on page 1055
IDSReportSet	Defines IDS action report set.	IDS	"IDSReportSet statement" on page 1057
IDSRule	Defines IDS rule.	IDS	"IDSRule statement" on page 1060

Table 65. Policy Agent configuration file policy statements (continued)

Statement	Purpose	Type	See
IDSScanEventCondition	Defines IDS rule scan event condition.	IDS	"IDSScanEventCondition statement" on page 1063
IDSScanExclusion	Defines IDS rule scan exclusion.	IDS	"IDSScanExclusion statement" on page 1065
IDSScanGlobalCondition	Defines IDS rule scan global condition.	IDS	"IDSScanGlobalCondition statement" on page 1067
IDSTRCondition	Defines IDS rule TR condition.	IDS	"IDSTRCondition statement" on page 1068
IpAddr	Defines IP address.	Reusable	"IpAddr statement" on page 1191
IpAddrGroup	Defines IP address group.	Reusable	"IpAddrGroup statement" on page 1192
IpAddrSet	Defines a single IP address or range of IP addresses.	Reusable	"IpAddrSet statement" on page 1193
IPDataOffer	Defines dynamic VPN data offer.	IPSec	"IpDataOffer statement" on page 1072
IPDynVpnAction	Defines IP filter dynamic VPN action.	IPSec	"IpDynVpnAction statement" on page 1077
IpFilterGroup	Defines IP filter policy group.	IPSec	"IpFilterGroup statement" on page 1083
IpFilterPolicy	Defines IP filter global policy information.	IPSec	"IpFilterPolicy statement" on page 1084
IPFilterRule	Defines IP filter policy rule.	IPSec	"IpFilterRule statement" on page 1087
IpGenericFilterAction	Defines IP filter generic action.	IPSec	"IpGenericFilterAction statement" on page 1091
IpLocalStartAction	Defines IP filter local start action.	IPSec	"IpLocalStartAction statement" on page 1093
IpManVpnAction	Defines IP filter manual VPN action.	IPSec	"IpManVpnAction statement" on page 1098
IpOptionGroup	Defines IP options group.	Reusable	"IpOptionGroup statement" on page 1195

Table 65. Policy Agent configuration file policy statements (continued)

Statement	Purpose	Type	See
IpOptionRange	Defines IP options.	Reusable	"IpOptionRange statement" on page 1195
IpProtocolGroup	Defines IP protocols group.	Reusable	"IpProtocolGroup statement" on page 1196
IpProtocolRange	Defines IP protocols.	Reusable	"IpProtocolRange statement" on page 1197
IpService	Defines IP filter rule service.	IPSec	"IpService statement" on page 1106
IpServiceGroup	Defines IP filter rule service group.	IPSec	"IpServiceGroup statement" on page 1111
IpTimeCondition	Defines time condition.	Reusable	"IpTimeCondition statement" on page 1198
Ipv6NextHdrGroup	Defines a group of IPv6 next header values	Reusable	"Ipv6NextHdrGroup statement" on page 1200
Ipv6NextHdrRange	Defines a range of IPv6 next header values	Reusable	"Ipv6NextHdrRange statement" on page 1200
KeyExchangeAction	Defines a key exchange action for a dynamic VPN.	IPSec	"KeyExchangeAction statement" on page 1111
KeyExchangeGroup	Defines a key exchange group.	IPSec	"KeyExchangeGroup statement" on page 1119
KeyExchangeOffer	Defines key exchange dynamic VPN offer.	IPSec	"KeyExchangeOffer statement" on page 1120
KeyExchangePolicy	Defines key exchange global policy information.	IPSec	"KeyExchangePolicy statement" on page 1126
KeyExchangeRule	Defines key exchange policy rule.	IPSec	"KeyExchangeRule statement" on page 1131
LocalDynVpnGroup	Defines local dynamic VPN policy group.	IPSec	"LocalDynVpnGroup statement" on page 1133
LocalDynVpnPolicy	Defines local dynamic VPN global policy information.	IPSec	"LocalDynVpnPolicy statement" on page 1134
LocalDynVpnRule	Defines local dynamic VPN policy rule.	IPSec	"LocalDynVpnRule statement" on page 1135

Table 65. Policy Agent configuration file policy statements (continued)

Statement	Purpose	Type	See
LocalSecurityEndpoint	Defines local security endpoint for IPSec policies.	IPSec	"LocalSecurityEndpoint statement" on page 1138
PolicyAction	Defines QoS policy action.	QoS	"PolicyAction statement" on page 1168
PolicyRule	Defines QoS policy rule.	QoS	"PolicyRule statement" on page 1176
PortGroup	Defines a port group.	Reusable	"PortGroup statement" on page 1201
PortRange	Defines a single port or range of ports.	Reusable	"PortRange statement" on page 1202
RemoteIdentity	Defines a single or wildcard value remote identity to use when negotiating dynamic VPN tunnels.	IPSec	"RemoteIdentity statement" on page 1144
RemoteSecurityEndpoint	Defines remote security endpoint for IPSec policies.	IPSec	"RemoteSecurityEndpoint statement" on page 1146
RouteTable	Defines Routing route table.	Routing	"RouteTable statement" on page 1152
RoutingAction	Defines Routing policy action.	Routing	"RoutingAction statement" on page 1163
RoutingRule	Defines Routing policy rule.	Routing	"RoutingRule statement" on page 1164
ServiceCategories	Defines V1 QoS policy action.	QoS	"ServiceCategories statement" on page 1183
ServicePolicyRules	Defines V1 QoS policy rule.	QoS	"ServicePolicyRules statement" on page 1187
TrafficDescriptor	Defines traffic descriptors.	Reusable	"TrafficDescriptor statement" on page 1203
TrafficDescriptorGroup	Defines traffic descriptor groups.	Reusable	"TrafficDescriptorGroup statement" on page 1205
TTLSCipherParms	Defines cipher specification for AT-TLS policies.	AT-TLS	"TTLSCipherParms statement" on page 1004

Table 65. Policy Agent configuration file policy statements (continued)

Statement	Purpose	Type	See
TTLSConnectionAction	Defines AT-TLS connection action.	AT-TLS	“TTLSConnectionAction statement” on page 1009
TTLSConnectionAdvancedParms	Defines AT-TLS advanced connection parameters.	AT-TLS	“TTLSConnectionAdvancedParms statement” on page 1012
TTLSEnvironmentAction	Defines AT-TLS environment action.	AT-TLS	“TTLSEnvironmentAction statement” on page 1017
TTLSEnvironmentAdvancedParms	Defines AT-TLS advanced environment parameters	AT-TLS	“TTLSEnvironmentAdvancedParms statement” on page 1020
TTLSGroupAction	Defines AT-TLS group action.	AT-TLS	“TTLSGroupAction statement” on page 1028
TTLSGroupAdvancedParms	Defines AT-TLS advanced group parameters.	AT-TLS	“TTLSGroupAdvancedParms statement” on page 1030
TTLSGskAdvancedParms	Defines AT-TLS System SSL advanced parameters.	AT-TLS	“TTLSGskAdvancedParms statement” on page 1032
TTLSGskLdapParms	Defines set of LDAP parameters for AT-TLS policies.	AT-TLS	“TTLSGskLdapParms statement” on page 1033
TTLSKeyringParms	Defines set of key ring parameters for AT-TLS policies.	AT-TLS	“TTLSKeyringParms statement” on page 1035
TTLSRule	Defines AT-TLS policy rule.	AT-TLS	“TTLSRule statement” on page 1036
TTLSignatureParms	Defines AT-TLS client elliptic curve preferences and signature algorithm pair specifications	AT-TLS	“TTLSSignatureParms statement” on page 1041

Rules:

- For statements of type QoS, policies are configured in the image or QoS image configuration file.
- For statements of type IDS, policies are configured in the common or image IDS configuration files.
- For statements of type IPSec, policies are configured in the common or image IPSec configuration files.
- For statements of type Routing, policies are configured in the common or image Routing configuration file.
- For statements of type AT-TLS, policies are configured in the common or image AT-TLS configuration files.
- For statements of type Reusable, policies are configured in the common or image IDS, IPSec, AT-TLS, or Routing configuration files.

AutoMonitorApps statement

You can configure the Policy Agent to monitor and automatically start or restart a set of related applications. The following set of applications can be monitored:
- Defense Manager daemon (DMD)
- IKE daemon (IKED)
- Network Security Server daemon (NSSD)
- Syslog daemon (SYSLOGD)
- Traffic Regulation Manager daemon (TRMD)

Use the AutoMonitorParms statement to configure global parameters that control how the Policy Agent monitors and starts or restarts these applications.

Use the AutoMonitorApps statement to configure which applications should be monitored and to specify application-specific parameters.

Restriction: To automatically monitor applications, Policy Agent must be started with a user ID that has superuser authority UID(0). For sample RACF commands, see the EZARACF member of SEZAINST.

Results:
- If you configure applications to be automatically started and restarted, be aware of the following results:
  - If you start the Policy Agent after you have already started an application to be monitored, Policy Agent starts monitoring the application (if it was originally started with the same job name that is configured to the Policy Agent). If the application needs to be restarted later, it is restarted using the cataloged procedure configured to the Policy Agent. This might not be the same procedure that was originally used to start the application.
  - If you start the Policy Agent after you have already started an application to be monitored, but the application does not use the same job name that is configured to the Policy Agent, the Policy Agent cannot detect that the application is active. Policy Agent tries to start another instance of the application, and this start is likely to fail.

    Tip: If you configure applications to be monitored by the Policy Agent, ensure that those applications are not running before you start the Policy Agent. Sometimes you might need to start syslogd before starting the Policy Agent. If you start syslogd before starting the Policy Agent, ensure that Policy Agent is configured with the correct syslogd job name.

- If this statement is removed, or one or more AppName parameters or instances of the TcpImageName parameter are removed, Policy Agent stops monitoring the affected applications. You must stop or restart the applications if needed.
- If one or more AppName parameters, or instances of the TcpImageName parameter are added, Policy Agent starts the affected applications and begins monitoring them.
- If any of the parameters other than AppName or TcpImageName are added, removed, or changed, Policy Agent stops and restarts the affected applications.

Syntax

```
►►──AutoMonitorApps──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{────────────────────────────────┬────────────────────────────────────────┤
   ├─┤ AutoMonitorApps Parameters ├───┤
   └─}────────────────────────────────┘
```

AutoMonitorApps Parameters:

```
    ┌──────────────────────────────────────────────────────────┐
    ▼                                                          │
├───┬──────────────────────────────────────────────────────────┴───────────────┤
    └─AppName──┬─DMD─────┬─┤ AppName Specification ├─┘
               ├─IKED────┤
               ├─NSSD────┤
               ├─SYSLOGD─┤
               └─TRMD────┘
```

AppName Specification:

```
├──┬─{───────────────────────────────────────┬─────────────────────────────────┤
   ├─┤ AppName Specification Parameters ├────┤
   ├─┤ MultiImage Specification ├────────────┤
   └─}───────────────────────────────────────┘
```

MultiImage Specification:

```
├──┬───────────────────────┬─┤ TcpImageName Specification ├────────────────────┤
   └─TcpImageName image────┘
```

TcpImageName Specification:

```
├──┬─{───────────────────────────────────────┬─────────────────────────────────┤
   ├─┤ AppName Specification Parameters ├────┤
   └─}───────────────────────────────────────┘
```

AppName Specification Parameters:

```
├──ProcName name──┬─────────────────┬──┬─────────────────────┬──────────────────►
                  └─Jobname name────┘  └─StartParms parms────┘

   ┌────────────────────────┐
   ▼                        │
►──┬────────────────────────┬┴──────────────────────────────────────────────────┤
   └─EnvVar envvar=value────┘
```

Parameters

AppName
: Specifies which applications you need to monitor and automatically start and restart. Repeat this parameter for each application.

TcpImageName
: A string 1 - 8 characters in length that specifies the TCP/IP images on which the application runs. Repeat this parameter for each image.

  Rules:

  - This parameter is required and valid only for applications that run a separate instance for each TCP/IP image. Currently, the only application that does this is TRMD.
  - You can specify a maximum of eight unique TcpImageName parameters for a given AppName parameter.
  - You must configure the specified TCP/IP image on a TcpImage statement.

  Results:

  - In a single stack (INET) environment, the application runs on the active TCP/IP image.
  - In a common INET (CINET) environment, if you do not specify the TCP/IP image name, the application runs on the default TCP/IP image (resolver supplied TCPIPuserid statement or TCPIPjobname statement). If the default TCP/IP image cannot be determined, the Policy Agent uses the name INET and the Policy agent creates an internal TcpImage statement with default values to represent the specified TCP/IP image.
  - If the TcpImage statement for the specified TcpImageName is removed, Policy Agent stops monitoring the application for that TCP/IP image.

ProcName
: A string 1 - 8 characters in length that specifies the name of a cataloged procedure that is used to start the application. A sample procedure is included in SEZAINST(EZAPOLPR).

  Tip: You can use a single generic cataloged procedure for all configured applications. Parameters are passed to the start procedure to identify the application name and application-specific parameters. If you use a single procedure, then all started applications run under the same user ID. If you want to use different user IDs for each application, specify different procedure names for the applications using this parameter.

  Rule: The specified procedure must contain the JCL parameters listed in Table 66 on page 953.

Table 66. JCL parameters

Procedure variable	Description	Value passed by the Policy Agent
PROG	Specifies the name of the application program executable.	One of the following supported application names: • DMD • IKED • NSSD • SYSLOGD • TRMD
VARS	Specifies the name of a temporary file containing environment variables for the application.	Temporary file name generated by the Policy Agent.
PARMS	Specifies start parameters for the application.	The string specified on the StartParms parameter on the AutoMonitorApps statement, or a null string.

Jobname
A string 1 - 8 characters in length that specifies the runtime job name for the application.

Rules:

- For applications that do not use the TcpImageName parameter, this parameter is optional. The default is the value specified with the AppName parameter.
- For applications that use the TcpImageName parameter, this parameter is required so that the job name for each instance is unique.

StartParms
A string 1 - 45 characters in length that specifies the start parameters for the application. Specify the parameters in the same way that you would specify them on the PARM parameter on the EXEC JCL statement, but do not include single quotation marks. For example:

```
StartParms   -d 1
```

EnvVar
A string 1 - 1 024 characters in length that specifies environment variables for the application. Repeat this parameter for each environment variable. Specify the environment variable name and the value, separated by an equal sign. The following examples show how this parameter can be used:

- To specify the configuration file for the IKED, use the following code:
  ```
  EnvVar   IKED_FILE=/etc/iked.conf
  ```
- To specify the resolver configuration file for the TRMD, use the following code:
  ```
  EnvVar   RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
  ```
- To specify the time zone for the NSSD, use the following code:
  ```
  EnvVar   TZ=EST5EDT
  ```

Examples

This example shows how to specify parameters for the following types of applications:

- An application without stack affinity, which means that a single copy of the application runs regardless of how many TCP/IP stacks are running. This example uses the IKED as such an application.
- An application with stack affinity, which means that one instance of the application runs on each TCP/IP stack. This example uses TRMD as such an application.

```
AutoMonitorApps
{
  AppName           IKED
  {
    Procname        POLPROC
  }
  AppName           TRMD
  {
    TcpImageName    TCPIP1
    {
      Procname      POLPROC
      Jobname       TRMD1
    }
    TcpImageName    TCPIP3
    {
      Procname      POLPROC
      Jobname       TRMD3
    }
  }
}
```

AutoMonitorParms statement

Use the AutoMonitorParms statement to configure the Policy Agent to monitor and automatically start or restart a set of related applications. The following set of applications can be monitored:

- Defense Manager daemon (DMD)
- IKE daemon (IKED)
- Network Security Server daemon (NSSD)
- Syslog daemon (SYSLOGD)
- Traffic Regulation Manager daemon (TRMD)

Use the AutoMonitorApps statement to configure what applications should be monitored and to specify application-specific parameters.

Use this statement to configure global parameters that control how the Policy Agent monitors and starts or restarts the configured applications. If the default values for all parameters are acceptable, you do not need to use this statement.

Results:

- If this statement is removed, the default values are applied when the previously specified MonitorInterval value expires.
- If this statement is added, the new values are applied when the previous default MonitorInterval value expires.
- If any changes are made to this statement, the new values are applied when the previously specified MonitorInterval value expires.

Syntax

```
►►──AutoMonitorParms──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬────────────────────────────────────┤
   ├─┤ AutoMonitorParms Parameters ├┤
   └─}──────────────────────────────┘
```

AutoMonitorParms Parameters:

```
   ┌─MonitorInterval 10─┐  ┌─RetryLimitCount 5─┐  ┌─RetryLimitPeriod 600─┐
├──┼────────────────────┼──┼───────────────────┼──┼──────────────────────┼────┤
   └─MonitorInterval n──┘  └─RetryLimitCount n─┘  └─RetryLimitPeriod n───┘
```

Parameters

`MonitorInterval`
Specifies the interval, in seconds, at which an application should be monitored to determine if that application is still running. Valid values are in the range 1 - 86400. The default value is 10 seconds.

`RetryLimitCount`
Specifies the number of times that Policy Agent should start or restart an application within the time period specified by the RetryLimitPeriod parameter. Valid values are in the range 1 - 99. The default value is 5.

Each time an application is started, Policy Agent waits 1 minute for the application to start. If it does not start, Policy Agent tries to start it again until the limit specified by this parameter is reached. If the application still has not started, Policy Agent stops monitoring the application until a MODIFY MON,START command is issued for the application. For example, if the application is the IKED, the MODIFY *procname*,MON,START,IKED command causes Policy Agent to resume trying to start the application. See MODIFY command: Policy Agent in z/OS Communications Server: IP System Administrator's Commands for information about using MODIFY commands to manage the monitored applications.

`RetryLimitPeriod`
Specifies the time interval, in seconds, at which Policy Agent should try to start or restart an application. See the RetryLimitCount parameter in this topic for more details. Valid values are in the range 1 - 86400. The default value is 600 (10 minutes).

ClientConnection statement

The Policy Agent acting as a policy server uses the ClientConnection statement to specify the listening port. The Policy Agent acting as a policy client uses this connection to retrieve remote policies.

Results:

- An error is flagged if both the ClientConnection and ServerConnection statements are configured on the same Policy Agent. The result is that there is no connection between the policy server and policy client.

- If the ClientConnection statement is removed, all connections to policy clients are disconnected.
- Updates to the ClientConnection statement are used only for new client connections to the policy server.

Syntax

```
                      ┌─16310─┐
►►──ClientConnection──┤       ├──────────────────────────────────────────►◄
                      └─port──┘
```

Parameters

port
Specifies the port that the policy server listens on for TCP connections from policy clients. This port must be the same as the ServerPort value specified on the ServerConnection statement for any policy clients that connect to this policy server.

The valid port values are in the range 1 - 65 535. The default port value is 16 310.

This statement is optional. If a ClientConnection statement is not configured, then the Policy Agent does not act as a policy server, and only listens for local connections using AF_UNIX sockets.

Result: If the port value is updated, then the policy server listens for TCP connections using the updated value.

Restriction: The port value cannot match the port value configured on the ServicesConnection statement.

Codepage statement

Use the Codepage statement to specify the EBCDIC code page to be used for reading all configuration files and policy definition files. The default is IBM-1047. All statements read from the files are converted to the IBM-1047 code page from the specified code page.

Result: If you specify a code page that is not one of the supported values, then Policy Agent issues a warning message to the log file and tries to read the configuration files using the IBM-1047 code page. It is possible that configuration errors might be detected in this case.

Syntax

```
►►──Codepage──codepage──────────────────────────────────────────────────►◄
```

Parameters

codepage
Specifies the EBCDIC code page to be used. The default code page is IBM-1047 if this statement is not specified. The following code pages are supported:
- IBM-037
- IBM-273
- IBM-274

- IBM-275
- IBM-277
- IBM-278
- IBM-280
- IBM-281
- IBM-282
- IBM-284
- IBM-285
- IBM-297
- IBM-500
- IBM-871
- IBM-1047
- IBM-1140
- IBM-1141
- IBM-1142
- IBM-1143
- IBM-1144
- IBM-1145
- IBM-1146
- IBM-1147
- IBM-1148
- IBM-1149

CommonIDSConfig statement

Use the CommonIDSConfig statement to specify the path of a local IDS policy file that contains common IDS policy statements. These common statements can be referenced from a stack-specific IDS policy file. To define a common set of policies for multiple stacks, use the IDSConfig statement to specify the same file as the CommonIDSConfig statement.

Stack-specific IDS policies are defined in a stack-specific IDS policy file. A stack-specific IDS policy file is identified by an IDSConfig statement.

The refresh interval for the CommonIDSConfig file is inherited from the main configuration file.

Specify the IDSConfig statement without a path name in each image configuration file to define a common set of policies for multiple stacks.

Restriction: The CommonIDSConfig statement can appear only in the main configuration file.

If a CommonIDSConfig statement appears multiple times in the main configuration file, the last occurrence of the statement is used. If the CommonIDSConfig statement appears in an image configuration file, it is ignored.

The configuration information defined in the file identified with the CommonIDSConfig statement is prepended to the configuration information defined in files identified with the IDSConfig statement. This action has the following consequences:

- If no IDSConfig statements are specified, then the CommonIDSConfig file is not parsed by Policy Agent.

  Requirement: The IDSConfig statement is required if IDS configuration files exist for a given stack.
- If multiple stacks are defined, the CommonIDSConfig file is parsed for each stack; thus, any errors contained in the file are reported multiple times.

Syntax

```
►►──CommonIDSConfig──path──────────────────────────────────────────────►◄
```

Parameters

path

The path of the common IDS policy file to be installed.

You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks ('') and must be preceded by a double slash (for example, //). The following examples show both types of names:

```
CommonIDSConfig  //'USER1.PAGENT.CONF(COMIDS)'
CommonIDSConfig  /u/user1/pagent.common.ids
```

Restriction: Dynamic monitoring for file updates using the -i startup option is supported only for file-system files; MVS data sets are not monitored for changes.

CommonIPSecConfig statement

Use the CommonIpSecConfig statement to specify the path of a local IPSec policy file that contains common IPSec policy statements. These common statements can be referenced from a stack-specific IPSec policy file. To define a common set of policies for multiple stacks, the IpSecConfig statement can specify the same file as the CommonIpSecConfig statement.

Stack-specific IPSec policies are defined in an IPSec stack-specific policy file. A stack-specific IPSec policy file is identified by an IpSecConfig statement. The refresh interval for the CommonIpSecConfig file is inherited from the main configuration file.

Specify the IPSecConfig statement without a path name in each image configuration file to define a common set of policies for multiple stacks.

Restriction: The CommonIpSecConfig statement can appear only in the main configuration file.

If a CommonIpSecConfig statement appears multiple times in the main configuration file, the last occurrence of the statement is used. If the CommonIpSecConfig statement appears in the image configuration file, it is ignored (unless the main and image configuration files are the same file).

The configuration information defined in the file identified with the CommonIPSecConfig statement is prepended to the configuration information defined in files identified with the IPSecConfig statement. This action has the following consequences:

- If no IPSecConfig statements are specified, then the CommonIPSecConfig file is not parsed by Policy Agent.

  Requirement: The IPSecConfig statement is required to define IPSec policy for a given stack.
- If multiple stacks are defined, the CommonIPSecConfig file is parsed for each stack; any errors contained in the file are reported multiple times.

Syntax

```
►►──CommonIpSecConfig──path──────────────────────────────────────────────►◄
```

Parameters

path

The path of the common IPSec policy file to be installed.

You can specify an MVS data set name or a UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). The following examples show both types of names:

```
CommonIPSecConfig  //'USER1.PAGENT.CONF(COMIPSEC)'
CommonIPSecConfig  /u/user1/pagent.common.ipsec
```

Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported only for z/OS UNIX files; MVS data sets are not monitored for changes.

CommonRoutingConfig statement

Use the CommonRoutingConfig statement to specify the path of a local Routing policy file that contains common Routing policy statements. These common statements can be referenced from a stack-specific Routing policy file. To define a common set of policies for multiple stacks, use the RoutingConfig statement to specify the same file as the CommonRoutingConfig statement.

Stack-specific Routing policies are defined in a stack-specific Routing policy file. A stack-specific Routing policy file is identified by a RoutingConfig statement.

The refresh interval for the CommonRoutingConfig file is inherited from the main configuration file.

Specify the RoutingConfig statement without a path name in each image configuration file to define a common set of policies for multiple stacks.

Restriction: The CommonRoutingConfig statement can appear only in the main configuration file.

If a CommonRoutingConfig statement appears multiple times in the main configuration file, the last occurrence of the statement is used. If the CommonRoutingConfig statement appears in an image configuration file, it is ignored.

The configuration information defined in the file identified with the CommonRoutingConfig statement is prepended to the configuration information defined in files identified with the RoutingConfig statement. This action has the following consequences:

- If no RoutingConfig statements are specified, then the CommonRoutingConfig file is not parsed by Policy Agent.

  Requirement: The RoutingConfig statement is required if Routing configuration files exist for a given stack.
- If multiple stacks are defined, the CommonRoutingConfig file is parsed for each stack; thus, any errors contained in the file are reported multiple times.

Syntax

```
►►──CommonRoutingConfig──path──────────────────────────────────────────►◄
```

Parameters

path

The path of the common Routing policy file to be installed.

You can specify an MVS data set name or a UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

```
CommonRoutingConfig  //'USER1.PAGENT.CONF(COMROUT)'
CommonRoutingConfig  /u/user1/pagent.common.routing
```

Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported for z/OS UNIX files only; MVS data sets are not monitored for changes.

CommonTTLSConfig statement

Use the CommonTTLSConfig statement to specify the path of a local AT-TLS policy file that contains common AT-TLS policy statements. These common statements can be referenced from a stack-specific AT-TLS policy file. To define a common set of policies for multiple stacks, the TTLSConfig statement can specify the same file as the CommonTTLSConfig statement.

Stack-specific AT-TLS policies are defined in a stack-specific AT-TLS policy file. A stack-specific AT-TLS policy file is identified by a TTLSConfig statement.

The refresh interval for the CommonTTLSConfig file is inherited from the main configuration file.

Specify the TTLSConfig statement without a path name in each image configuration file to define a common set of policies for multiple stacks.

Restriction: The CommonTTLSConfig statement can appear only in the main configuration file.

If a CommonTTLSConfig statement appears multiple times in the main configuration file, the last occurrence of the statement is used. If the CommonTTLSConfig statement appears in an image configuration file, it is ignored (unless the main and image configuration files are the same file).

The configuration information defined in the file identified with the CommonTTLSConfig statement is prepended to the configuration information defined in files identified with the TTLSConfig statement. This action has the following consequences:

- If no TTLSConfig statements are specified, then the CommonTTLSConfig file is not parsed by Policy Agent.

  Requirement: The TTLSConfig statement is required to define AT-TLS policy for a given stack.
- If multiple stacks are defined, the CommonTTLSConfig file is parsed for each stack, so any errors contained in the file are reported multiple times.

Syntax

```
►►──CommonTTLSConfig──path──────────────────────────────────────────►◄
```

Parameters

path

The path of the common AT-TLS policy file to be installed.

You can specify an MVS data set name or a UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

```
CommonTTLSConfig  //'USER1.PAGENT.CONF(COMTTLS)'
CommonTTLSConfig  /u/user1/pagent.common.ttls
```

Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported for z/OS UNIX files only; MVS data sets are not monitored for changes.

DynamicConfigPolicyLoad statement

The Policy Agent acting as a policy server uses the DynamicConfigPolicyLoad statement to obtain the file names of the configuration files to be retrieved by policy clients.

The DynamicConfigPolicyLoad statement can appear only in the main configuration file.

For each policy type the policy client files are read only after the policy client connects to the policy server. A DynamicConfigPolicyLoad statement (or default values) is bound to a policy client for the life of that client, until one of the following occurs:

- The policy client disconnects from the policy server.
- The connection between the policy server and policy client ends.
- The associated DynamicConfigPolicyLoad statement is removed.

Result: When a DynamicConfigPolicyLoad statement is removed, the policy clients that were using that statement change to use another statement, or default values.

The policy client policies are removed from the policy server in the following cases:

- The policy client disconnects from the policy server.
- The connection between the policy server and the policy client ends.

- The policy client requests that remote policies for a specific policy type be unloaded from the policy server.
- The associated DynamicConfigPolicyLoad statement is removed.

To retrieve remote policies with the policy client, you must define security product authority in the SERVAUTH class for the policy client's user ID; the user ID is defined on the Userid parameter on the PolicyServer statement. The ClientName parameter on the PolicyServer statement is used as the image name. For more information, see the general policy agent configuration information in z/OS Communications Server: IP Configuration Guide. Wildcard values are allowed in profile names. The following example shows the structure of the security product profile:

```
EZB.PAGENT.sysname.image.ptype
```

Multiple DynamicConfigPolicyLoad statements can appear in the main configuration file. The policy server maintains a list of these DynamicConfigPolicyLoad statements. When a policy client connects to the policy server, then the policy client name configured on the PolicyServer statement is matched to the *clientname* parameter. The names are case sensitive with regard to matching. This *clientname* parameter can be a regular expression. The policy server matches these names in the following order:

1. A *clientname* parameter that has an exact match to the policy client name. The policy client name must not contain any regular expression characters.
2. A regular expression *clientname* parameter that matches the policy client name. The longest matching regular expression is chosen. If multiple statements match with the same length *clientname* parameter, the statement chosen is based on alphabetical order.
3. If there is no matching *clientname* parameter or a matching *clientname* value does not have a corresponding PolicyType parameter for this policy type, then the following default remote files are used:
   - Stack-specific remote files used for each policy type:
     - IDS - /etc/pagent_remote.ids
     - IPSec - /etc/pagent_remote.ipsec
     - QoS - /etc/pagent_remote.qos
     - Routing - /etc/pagent_remote.routing
     - AT-TLS - /etc/pagent_remote.ttls
   - For any default stack-specific remote file used, there is no corresponding common configuration file.
   - If no matching *clientname* parameter is found, then the refresh interval is set to 30 minutes.

Result: The PolicyLoad and CommonPolicyLoad parameters are optional; however, if neither the PolicyLoad parameter or the CommonPolicyLoad parameters are configured, this DynamicConfigPolicyLoad statement results in an error and the statement is discarded.

Syntax

```
►►──DynamicConfigPolicyLoad──clientname──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────────────┬────────────────────────────────────────────┤
   ├┤ DynamicConfigPolicyLoad Parameters ├─┤
   └─}─────────────────────────────────────┘
```

DynamicConfigPolicyLoad Parameters:

```
   ┌──────────────────────────────────────────────────────────┐
   ▼                                                          │
├──┴─┬─────────────────────────────────────────────────────┬──┴──────────────────────►
     └─PolicyType──┬─IDS─────┬┤ PolicyTypeSpecification ├─┘
                   ├─IPSec───┤
                   ├─QoS─────┤
                   ├─Routing─┤
                   └─TTLS────┘

  ┌─RefreshInterval 1800─┐
►─┴─RefreshInterval i────┴───────────────────────────────────────────────────────────┤
```

PolicyTypeSpecification:

```
├──┬─{─────────────────────────────────────┬────────────────────────────────────────────┤
   ├┤ PolicyTypeSpecification Parameters ├─┤
   └─}─────────────────────────────────────┘
```

PolicyTypeSpecification Parameters:

```
├──┬─────────────────────────┬──┬──────────────────┬────────────────────────────────────┤
   └─CommonPolicyLoad path─┘  └─PolicyLoad path─┘
```

Parameters

`clientname`

A string 1 - 24 characters in length specifying the client name to be matched to the policy client name.

Requirement: If this is a regular expression, the string must consist of 1 - 511 characters. Otherwise, it must consist of 1-24 characters.

This *clientname* parameter is used to match the policy client name when it connects to Policy Agent to derive its policy files.

The *clientname* parameter can also consist of a regular expression. The simplest form of regular expression is a string of characters without a special meaning. Such a string matches only itself. Table 67 shows the characters with special meaning:

Table 67. Characters with special meaning

Symbol	Description
.	The period symbol matches any one character except the terminal newline character.

Table 67. Characters with special meaning (continued)

Symbol	Description
[character–character]	The hyphen symbol (-), within square brackets, means through. It fills in the intervening characters according to the current collating sequence. For example, [a–z] can be equivalent to [abc...xyz] or, with a different collating sequence, it can be equivalent to [aAbBcC...xXyYzZ].
[string]	A string within square brackets specifies any of the characters in the string. Thus [abc], if compared to other strings, matches any that contain a, b, or c.
[m] [m,] [m,u]	Integer values enclosed within square brackets indicate the number of times to apply the preceding regular expression. The m value is the minimum number, and the u value is the maximum number. The u value must be less than 256. If you specify only the m value, it indicates the exact number of times to apply the regular expression. [m,] is equivalent to [m,u]. They both match m or more occurrences of the expression. The plus (+) and asterisk (*) operations are equivalent to [1,] and [0,], respectively.
*	The asterisk (*) indicates 0 or more of any characters. For example, [a*e] matches any of the following: 99ae9, aaaaae, or a999e99.
^	The caret symbol matches the beginning of the string.
$	The dollar symbol matches the end of the string. (Use \n to match a newline character.)
+	The plus symbol specifies one or more occurrences of a character. Thus, smith+ern is equivalent to smithhhern.
[^string]	The caret symbol inside square brackets, negates the characters within the square brackets. Thus [^abc] matches any characters except a, b, or c.
(expression)	Groups a sub-expression allowing an operator, such as * or +, to work on the sub-expression enclosed in parentheses. For example, (a*(cb+)*).

Rules:

- Do not use multibyte characters.
- You can use the right square bracket ( ]) alone within a pair of square brackets, but only if it immediately follows either the opening left square bracket or if it immediately follows [^. For example, []–] matches the] and – characters.
- All the preceding symbols are special. Precede them with a slash (\) to use the symbol itself. For example, a\.e is equivalent to a.e.

- You can use the hyphen (-) by itself, but only if it is the first or last character in the expression. For example, the expression []--0] matches either the] or else the characters – through 0. Otherwise, use \–.
- If duplicate DynamicConfigPolicyLoad statements with the same *clientname* parameter are specified, Policy Agent keeps the last entry.
- If a matching DynamicConfigPolicyLoad statement cannot be found, then the default stack-specific remote policy file is used.
- You cannot specify duplicate symbolic values in a single file name. For example, you cannot specify /etc/$1.$2_$1.
- You cannot use both wildcard and symbolic values in the same file name. For example, you cannot use /etc/$1.* .

PolicyType
Indicates additional policy configuration information for a specific policy type.

Rule: If duplicate PolicyType parameters for the same policy type are configured, then the policy server keeps the last entry for that policy type.

RefreshInterval
Specifies the time interval (in seconds) that lapses between checks for changes to the creation or modification time of the common and stack-specific remote policy files. This attribute applies to all configured policy types. In the following cases, the update interval is changed:

- If a value is not specified, the default is 1 800 seconds (30 minutes).
- If a value of 0 is specified, the default value of 1 800 seconds (30 minutes) is used.
- Any value from 1 to 299 is rounded up to 300 seconds (5 minutes).

For example, if the refresh interval is set to 300, the corresponding policy file is checked for changes every five minutes. If the policy file changed within the last 5 minutes, it is read again. Any new, changed, or deleted policies are either added to or removed from the policy client configuration.

Result: If the RefreshInterval parameter is updated, this new refresh interval takes effect the next time these policies are refreshed.

Restriction: Dynamic monitoring for file updates using the **-i** startup option is not supported for files configured on the DynamicConfigPolicyLoad statement.

CommonPolicyLoad
The path of the common remote policy file to be used for the defined policy type.

You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

```
DynamicConfigPolicyLoad (.*)_(.*)
{
PolicyType  IDS
{
CommonPolicyLoad  //'USER1.PAGENT.REMCONF(COMIDS)'
PolicyLoad        //'USER1.PAGENT.REMCONF(IDS)'
}
PolicyType  TTLS
{
CommonPolicyLoad  /u/user1/pagent.remote.common.ttls
PolicyLoad        /u/user1/pagent.remote.ttls
}
}
```

The common remote policy file statements can be referenced from the stack-specific remote policy file of the associated policy configuration. Stack-specific remote policies are defined in the stack-specific remote policy file within the same policy configuration. A stack-specific remote policy file is identified by the PolicyLoad parameter.

The configuration information defined in the file identified with the CommonPolicyLoad parameter is prepended to the configuration information defined in the file identified with the PolicyLoad parameter.

Rule: If the DynamicConfigPolicyLoad statement matches multiple policy clients, then the CommonPolicyLoad file is parsed for each policy client. Any errors contained in the file are reported multiple times.

Restrictions:

- Dynamic monitoring for file updates using the**-i**startup option is not supported for the common remote policy file.
- The CommonPolicyLoad parameter is not supported for PolicyType QoS.

Results:

- When the common remote policy file is an MVS data set, it is reread at each refresh interval, regardless of whether it has actually been changed or not. The policy server also rereads all the associated stack-specific remote policy files when the common remote policy file is reread.
- If the CommonPolicyLoad parameter file name is updated, this new common file is read when policies are refreshed.

PolicyLoad

The path of the stack-specific remote policy file to be used for the defined policy type.

You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

```
DynamicConfigPolicyLoad (.*)_(.*)
{
PolicyType  IDS
{
CommonPolicyLoad  //'USER1.PAGENT.REMCONF(COMIDS)'
PolicyLoad        //'USER1.PAGENT.REMCONF(IDS)'
}
PolicyType  TTLS
{
CommonPolicyLoad  /u/user1/pagent.remote.common.ttls
PolicyLoad        /u/user1/pagent.remote.ttls
}
}
```

Rules:

- If the PolicyLoad parameter is not specified, then the associated common remote policy file specified on the CommonPolicyLoad parameter is used.
- The client names and DynamicConfigPolicyLoad statement names are case sensitive, but MVS data set names are not. Therefore, use caution when defining MVS data set configuration files that include a wildcard to be substituted with the client name. For example, the client names client42 and Client42, if used as a substitution variable in an MVS data set name, would result in the same configuration file being used for both clients.

Results:

- The path name can contain a single wildcard character (*). The policy client name replaces the wildcard position to obtain the stack-specific remote policy file.

  The following examples use a wildcard path for an IPSec file:

  ```
  PolicyLoad    //'ETC.REMOTE.CONF(*)'
  policy client name = Remote1
  Stack-specific remote IPSec policy file is:
  //'ETC.REMOTE.CONF(REMOTE1)'
  ```

  ```
  PolicyLoad    /etc/*.remote
  policy client name = REMOTE1
  Stack-specific remote IPSec policy file is:
  /etc/REMOTE1.remote
  ```

- The path name can contain symbolic replacement values $0 through $9. $0 represents the entire portion of the client name that matched, while $1 through $9 represent portions of the client name that match corresponding parenthesized sub-expressions in the regular expression.

  Example of using symbolic replacement values for an IDS file:

  ```
  Regular expression = ^([A-Z].+[a-z]+)\.([A-Z].+[a-z]+)$
  PolicyLoad    //'ETC.$1($2)'

  policy client name = SYSa.IDSClient
  Stack-specific remote IDS policy file will be: //'ETC.SYSA(IDSCLIENT)
  ```

  Result: If more symbolic replacement values are specified in a file name than there are parenthesized sub-expressions in the regular expression, the extra symbolic replacement values are not replaced and exist as literal values in the file name.

Restriction: Dynamic monitoring for file updates using the **-i**startup option is not supported for the stack-specific remote policy file.

Results:

- When the stack-specific remote policy file is an MVS data set, it is reread at each refresh interval, regardless of whether it has actually been changed or not.
- If the PolicyLoad parameter file name is updated, the new stack-specific file is read when policies are refreshed.

IDSConfig statement

Use the IDSConfig statement to specify the path of a local IDS policy file that contains stack-specific IDS policy statements.

Requirement: The IDSConfig statement is required to define IDS configuration file policy for a given stack.

Specify the IDSConfig statement without a path name in each image configuration file to define a common set of policies for multiple stacks.

Results: For the associated TCP/IP image on the policy client, if the PolicyServer statement specifies remote IDS policies, then the following occurs:

- If no local IDS policies are installed, then the IDSConfig statement is ignored.
- If local IDS policies are already installed, the result is the same as if the IDSConfig statement had been deleted.

Use the FLUSH/NOFLUSH and PURGE/NOPURGE parameters to specify whether or not IDS policies are deleted at startup (and when a MODIFY PAGENT,REFRESH command is issued) and shutdown, respectively.

The refresh interval for the IDSConfig file is inherited from the image configuration file containing the corresponding IDSConfig statement.

Restriction: The IDSConfig statement can appear only in an image configuration file.

If an IDSConfig statement appears multiple times in an image configuration file, the last occurrence of the statement is used. If the IDSConfig statement appears in the main configuration file, it is ignored (unless the main and image configuration files are the same file).

Syntax

```
►►──IDSConfig──┬──────┬──┬─────────┬──┬─────────┬──────────────────────────────►◄
               └─path─┘  ├─FLUSH───┤  ├─PURGE───┤
                         └─NOFLUSH─┘  └─NOPURGE─┘
```

Parameters

path
: Specifies the path of the stack-specific IDS policy file to be installed. If no path name is specified, the common IDS policy file specified on the CommonIDSConfig statement is used.

  You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

  ```
  IDSConfig  //'USER1.PAGENT.CONF(IDS)'
  IDSConfig  /u/user1/pagent.ids
  ```

  Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported only for file-system files; MVS data sets are not monitored for changes.

FLUSH
: Specifies that all policies installed in the Policy Agent and the TCP/IP stack are deleted. Policies are flushed when the following occurs:
  - A new TcpImage statement is processed for the first time, including Policy Agent starting
  - A MODIFY PAGENT,REFRESH command is entered

NOFLUSH
: Specifies that all policies installed in the Policy Agent and the TCP/IP stack are to remain during initial startup and at each refresh interval. In addition, policies that are deleted from a configuration are not deleted from the Policy Agent or the TCP/IP stack.

PURGE
: Specifies that all policies installed in the TCP/IP stack are deleted during normal termination and when a TcpImage or PEPInstance statement is deleted.

NOPURGE
Specifies that no policies that are installed in the TCP/IP stack are deleted during normal termination and when a TcpImage or PEPInstance statement is deleted.

For details, see the FLUSH and PURGE information in z/OS Communications Server: IP Configuration Guide.

Result: If the IDSConfig statement is deleted and the FLUSH parameter is configured, then all IDS configuration file policies are deleted from the corresponding stack.

IPSecConfig statement

Use the IpSecConfig statement to specify the path of a local IPSec policy file that contains stack-specific IPSec policy statements.

Specify the IPSecConfig statement without a path name in each image configuration file to define a common set of policies for multiple stacks.

Requirement: The IpSecConfig statement is required to define IPSec policy for a given stack.

The refresh interval for the IpSecConfig file is inherited from the image configuration file containing the corresponding IpSecConfig statement.

Results: For the associated TCP/IP image on the policy client, if the PolicyServer statement specifies remote IPSec policies, then the following occurs:
- If no local IPSec policies are installed, then the IPSecConfig statement is ignored.
- If local IPSec policies are already installed, the result is the same as if the IPSecConfig statement had been deleted.

Rule: For IPSec policies, when errors are detected during parsing, no new policies are installed.

The IpSecConfig statement can appear only in an image configuration file. If an IpSecConfig statement appears multiple times in an image configuration file, the last occurrence of the statement is used. If the IpSecConfig statement appears in the main configuration file, it is ignored (unless the main and image configuration files are the same file).

Syntax

```
►►──IPSecConfig──┬──────┬──────────────────────────────────────────►◄
                 └─path─┘
```

Parameters

path
The path of the stack-specific IPSec policy file to be installed. If no path name is specified, then the common IPSec policy file specified on the CommonIpSecConfig statement is used.

You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

```
IPSecConfig  //'USER1.PAGENT.CONF(IPSEC)'
IPSecConfig  /u/user1/pagent.ipsec
```

Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported only for z/OS UNIX files; MVS data sets are not monitored for change.

Result: If the IPSecConfig statement is deleted, then all IPSec policies are deleted from the corresponding stack. The stack reverts to using the filter policy defined using the IPSEC statement in the TCP/IP profile. All IPSec policies for the stack are deleted from IKE.

LogLevel statement

Use the LogLevel statement to specify the level of tracing for the Policy Agent. use the trace records to help debug errors in policy definition.

Syntax

```
►►──LogLevel──i────────────────────────────────────────────────────────►◄
```

Parameters

i

An integer that specifies the level of logging and tracing. The following levels are supported:

- 1 - SYSERR - System error messages
- 2 - OBJERR - Object error messages
- 4 - PROTERR - Protocol error messages
- 8 - WARNING - Warning messages
- 16 - EVENT - Event messages
- 32 - ACTION - Action messages
- 64 - INFO - Informational messages
- 128 - ACNTING - Accounting messages
- 256 - TRACE - Trace messages

Usage notes

Use this statement to specify a desired log level or a combination of levels. If this statement is absent, the default level is 31.

To combine log levels, add log level numbers. For example, to request SYSERR messages (level 1) and EVENT messages (level 16), request log level 17.

Examples

The following example turns on all trace levels for Policy Agent:

```
LogLevel 511
```

PolicyPerfMonitorForSDR statement

Use the PolicyPerfMonitorForSDR statement to enable or disable the policy performance monitor function. This function assigns weight fractions to the monitored policy performance data and sends them to the sysplex distributor (SD) distributing stack as the monitored data crosses defined thresholds. The SD distributing stack uses these weight fractions to influence its routing decisions for incoming connection requests toward appropriate hosts within a group responsible for processing the requests. These connection requests are for a specific application (for example, HTTP web) for which one or more policies have been defined. For more information about sysplex distributor policy performance monitoring, see z/OS Communications Server: IP Configuration Guide.

Restriction: This statement applies only when policies are defined for the TCP protocol.

Syntax

```
►►──PolicyPerfMonitorForSDR──┬─Enable──┬─┤ Put Braces and Parameters on Separate Lines ├─────────►◄
                             └─Disable─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────────────┬────────────────────────────────────────────┤
   ├─┤ PolicyPerfMonitorForSDR Parameters ├─┤
   └─}─────────────────────────────────────┘
```

PolicyPerfMonitorForSDR Parameters:

```
   ┌─SamplingInterval ──60─┐  ┌─LossRatioAndWeightFr ──10 10──┐
├──┼───────────────────────┼──┼───────────────────────────────┼──────────────────────►
   └─SamplingInterval ──n──┘  └─LossRatioAndWeightFr──┬─r f─┬─┘
                                                      └─0───┘

   ┌─LossMaxWeightFr ──100─┐  ┌─TimeoutRatioAndWeightFr ──10 20──┐
►──┼───────────────────────┼──┼──────────────────────────────────┼──────────────────►
   └─LossMaxWeightFr ──n───┘  └─TimeoutRatioAndWeightFr──┬─r f─┬─┘
                                                         └─0───┘

   ┌─TimeoutMaxWeightFr ──100─┐  ┌─ MaxConnWeightFr ──70 85 95─┐
►──┼──────────────────────────┼──┼─────────────────────────────┼────────────────────┤
   └─TimeoutMaxWeightFr ──n───┘  └─MaxConnWeightFr ──n n n─────┘
```

Parameters

`Enable | Disable`

Enables or disables the policy performance monitor function. When active, this function monitors policy performance data on sysplex distributor target stacks and sends information to the Sysplex Distributor distributing stack to be used in balancing the workload among the target stacks. The policy performance data is based on statistics for traffic that maps to defined service policies.

The weight fractions determined for the loss ratio and timeout ratio are added together to form a single weight fraction before being sent to the SD distributing stack. One weight fraction is generated for each DVIPA/port pair on SD target stacks that have at least one policy defined that maps to traffic sent from the target DVIPA/port pair.

SamplingInterval
Specifies the interval in seconds for sampling policy performance data. The default is 60.

LossRatioAndWeightFr
Specifies two numbers. The first is the unit ratio of retransmitted bytes (loss) over transmitted bytes, in tenths of a percent (1 - 1 000). The second number is the weight fraction to be returned to the sysplex distributor distributing stack, in percentage (1 - 100). When present, this parameter results in creation of a threshold table. The first number defines the loss ratio initial threshold value. The second number defines the starting weight fraction that the sysplex distributor distributing stack is to use to reduce the WLM weight for this target stack. For example, if the weight fraction is 50% and the WLM weight is 64, then the resulting weight used for this target stack is 32. The LossMaxWeightFr parameter determines the maximum weight fraction that is reached. The default values for each number is 10. A weight fraction of 0 instructs the system to suppress the loss ratio factor in sysplex distributor computations.

Use the following formula to calculate the threshold table:

```
if  x(n)% <= % Packet Loss  < x(n+1)%, then weight fraction is y(n)%
```

x	Initial loss ratio percentage (first number)
y	Initial weight fraction (second number)
n	Integer multiplier

For example, if the first and second numbers are 10 and 10, then the threshold table is:

```
n=0 : 0% <= % packet loss < 1%; weight fraction is 0%
n=1 : 1% <= % packet loss < 2%; weight fraction is 10%
n=2 : 2% <= % packet loss < 3%; weight fraction is 20%
n=3 : 3% <= % packet loss < 4%; weight fraction is 30%
.
.
.
1(n)%  <= % packet loss < 1(n+1)%; weight fraction is 10(n)%
```

If the first and second numbers are 30 and 20, then the threshold table is:

```
n=0: 0% <= % packet loss < 3%; weight fraction is 0%
n=1: 3% <= % packet loss < 6%; weight fraction is 20%
n=2: 6% <= % packet loss < 9%; weight fraction is 40%
n=3: 9% <= % packet loss < 12%; weight fraction is 60%
.
.
.
3(n)%  <= % packet loss < 3(n)%; weight fraction is 20(n)%
```

Tip: These ratios are not only used as input to create the these weight fractions, but are also used to create the service level fractions. See z/OS Communications Server: IP Configuration Guide for more information about policy based networking.

LossMaxWeightFr
Specifies the maximum weight fraction to be assigned for the loss ratio factor. The default is 100 %.

TimeoutRatioAndWeightFr

Specifies two numbers. The first number is the unit ratio of the number of timeouts over transmitted packets, in tenths of a percent (1 - 1 000). The second number is the weight fraction to be returned to the sysplex distributor distributing stack, in percentage (1 - 100). When present, this parameter results in a creation of a threshold table. The first number defines the timeout ratio initial threshold value. The second number defines the starting weight fraction that the sysplex distributor distributing stack is to use to reduce the WLM weight for this target stack. For example, if the weight fraction is 50% and the WLM weight is 64, the resulting weight used for this target stack is 32. The maximum weight fraction reached is determined by the TimeoutMaxWeightFr parameter. The default values are 10 and 20. A weight fraction of 0 instructs the system to suppress the timeout ratio factor in sysplex distributor computations. See the LossRatioAndWeightFr parameter for more information about how the threshold table is calculated.

TimeoutMaxWeightFr

Specifies the maximum weight fraction to be assigned for the timeout ratio factor. The default is 100%.

MaxConnWeightFr

Specifies three percentages that are used in calculating the connection limit portion of the policy action (service level) weight fractions.

Restriction: Each percentage must be in the range 1 - 100, and each value must be greater than or equal to the preceding value. The default values are 70, 85, and 95. When calculating the policy action weight fraction, the number of active connections to a target DVIPA/Port is compared with the maximum connections allowed for the associated policy action as follows:

- When the number of active connections reaches the percentage of maximum connections specified by the first number, the policy action weight fraction is set to MAX (50%, current calculated value).
- When the number of active connections reaches the percentage of maximum connections specified by the second number, the Policy Action weight fraction is set to MAX (85%, current calculated value).
- When the number of active connections reaches the percentage of maximum connections specified by the third number, the Policy Action weight fraction is set to 100%.

For more information about how the Policy Agent calculates policy action weight fractions, see z/OS Communications Server: IP Configuration Guide.

Examples

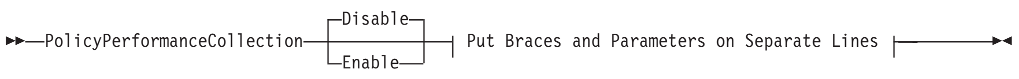
In this example, Policy Agent sends a message to the SD distributing Stack when the loss (retransmission) ratio begins to exceed 1% but not above 2%, with a weight fraction of 20% . This means that the WLM weight is reduced by 20% before it is used as a measure to route incoming connection requests. When the loss (retransmission) ratio exceeds 2%, but not above 3%, a message is sent with a weight fraction of 40%, and so on. When the loss exceeds 5%, a maximum weight fraction of 100% is used. The same is true with the timeout ratio. When the timeout ratio exceeds 0.5%, but not above 1%, a weight fraction of 50% is added to the weight in the message sent to the SD distributing Stack, and so on.

```
PolicyPerfMonitorForSDR    Enable
{
 SamplingInterval            120
    LossRatioAndWeightFr       10    20
    LossMaxWeightFr                100
    TimeoutRatioAndWeightFr     5   50
    TimeoutMaxWeightFr             100
}
```

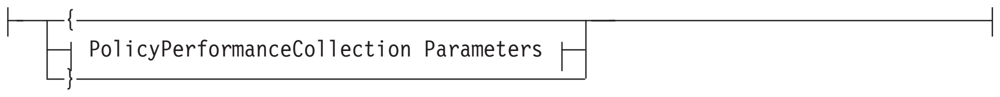
PolicyPerformanceCollection statement

Use the PolicyPerformanceCollection statement to enable or disable the policy performance collection function. Use this function to collect QoS performance monitoring data. The performance data can be collected on a policy rule or action or on both rules and actions. The collected data can also be logged to a specified performance log file for offline collection and monitoring by a user application, or can be accessed in near real time using the Policy API (PAPI).

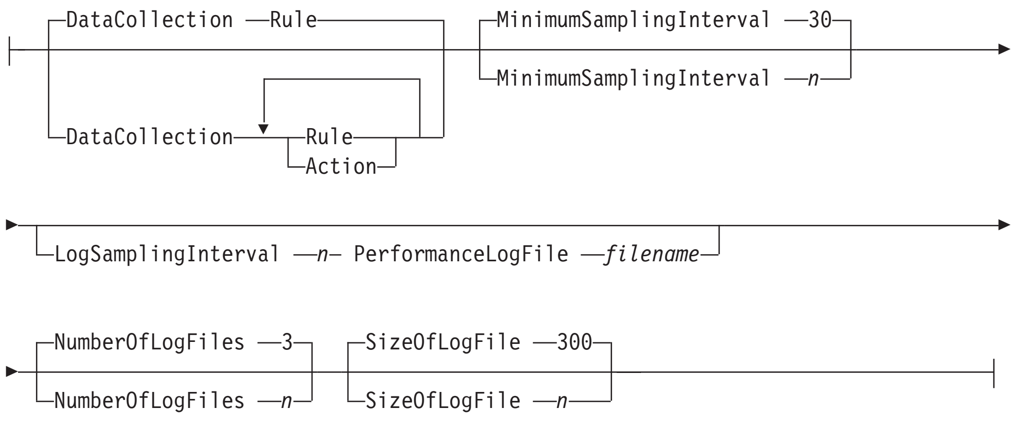
Syntax


PolicyPerformanceCollection
Disable
Enable
Put Braces and Parameters on Separate Lines

Put Braces and Parameters on Separate Lines:



PolicyPerformanceCollection Parameters:



Parameters

`Enable | Disable`
Enables or disables policy performance collection. When active, this function collects the QoS performance data from the stack. The default is Disable.

`DataCollection`
Specifies the type of performance data that needs to be collected. The accepted values are:

Rule To collect performance information for rules

Action
To collect performance information for actions.

Tip: Any combination of Rule, Action, or Rule Action can be used. If multiple types are used, separate them with a space (for example, Rule Action). The default value is Rule. The information returned for a policy action is an aggregate of the information for all the policy rules that use that policy action.

When FLUSH is specified on the TcpImage statement that defines the stack that is collecting performance data, the performance metrics are reset to 0 at the following times:

- When a new TcpImage statement is processed for the first time, including Policy Agent starting
- When a MODIFY PAGENT,REFRESH command is entered

If NOFLUSH is specified, the performance metrics are not reset.

`MinimumSamplingInterval`
Specifies the minimum sampling interval (in seconds) at which the performance data is retrieved from the stack. If the client, using the PAPI papi_get_perf_data() function, specified the acceptableCachedTime that is smaller than this value, the acceptableCachedTime value is overridden by MinimumSamplingInterval. See z/OS Communications Server: IP Programmer's Guide and Reference for more information about PAPI. The default value is 30 seconds. The values for MinimumSamplingInterval are in the range 30 - 2 147 483 647.

`LogSamplingInterval`
Specifies the log sampling interval in seconds at which the performance data is retrieved from the stack and logged into the log file defined by PerformanceLogFile parameter. The values for LogSamplingInterval are in the range 30 - 2 147 483 647.

Restriction: If LogSamplingInterval and PerformanceLogFile are not both specified, Policy Agent does not log the performance information.

`PerformanceLogFile`
Specifies the name of the file where collected performance data is written.

Restriction: If LogSamplingInterval and PerformanceLogFile are not both specified, Policy Agent does not log the performance information. This must be a z/OS UNIX file name. The file is created if it does not exist.

The TcpImage name for the stack for which this statement is configured is appended to the log file name, along with a numeric digit when NumberOfLogFiles is greater than 1. The format of the file name is `PerformanceLogFile.TcpImage.n`

For example, if PerformanceLogFile specified /u/user10/perflog, NumberOfLogFiles is 2, and TcpImage is TCPCS, the files is named:

```
/u/user10/perflog.TCPCS
/u/user10/perflog.TCPCS.1
```

The data in the log file is in binary format. The format is as follows. See the descriptions of the output from the NETSTAT SLAP or **netstat -j** report in z/OS Communications Server: IP System Administrator's Commands for more detail on the meaning of the various fields. Also, see the Network SLAPM2 MIB, shipped as a sample file, for additional information about the performance data.

- 4–byte time stamp in time_t format, as output by the C currentTime() function, when the record was created

- 4–byte version identifier, as defined by PCOL_LOG_VERSION in the papiuser.h header file
- 48–byte policy name
- 4–byte record type
- 4–byte record ID
- 4–byte time stamp in time_t format, when the policy was last activated
- 4–byte time stamp in time_t format, when the policy was last mapped to any traffic
- 8–byte count of total bytes transmitted
- 8–byte count of total packets transmitted
- 4–byte count of active connections
- 4–byte reserved field
- 8–byte count of total accepted connections
- 4–byte average smoothed TCP round trip time (RTT)
- 4–byte mean deviation of smoothed TCP RTT
- 8– byte count of total bytes retransmitted
- 8–byte count of total packets retransmitted
- 4–byte average smoothed TCP connection delay
- 4–byte mean deviation of smoothed TCP connection delay
- 4–byte average TCP accept queue delay
- 4–byte mean deviation of TCP accept queue delay
- 8–byte count of total packets transmitted in profile
- 8–byte count of total bytes transmitted in profile
- 16–byte reserved field network slpa
- 8–byte count of total packets received
- 8–byte count of total bytes received
- 8–byte count of total retransmitted packets timed out
- 8–byte count of total denied connections
- 24–byte reserved field

NumberOfLogFiles

Specifies the number of performance log files to be maintained. The default value is 3. The values for NumberOfLogFiles are in the range 1 - 255. The log files are maintained in a round-robin fashion. When the current log file fills up, a new log file is created and all existing log files are renamed, with the oldest file being deleted if the total number of files would exceed the NumberOfLogFiles parameter. Each renamed file has a numeric digit added to the end of the name.

SizeOfLogFile

Specifies the log file size in kilobytes (Kb). The default value is 300 Kb. The values for SizeOfLogFile are in the range 1 Kb - 1 000 000 Kb.

The amount of data that fits in the log files, and therefore the amount of time that elapses before the files wrap, depends on a number of factors. Each performance data record is 232 bytes in length. You can use the following formulas:

- Size of log file in bytes * number of log files / 232 = number of records
- Number of records / number of policies = number of refresh cycles

- Number of refresh cycles * refresh interval in minutes = minutes worth of data

For example, assume 5 log files with a size of 400 kilobytes. Also, assume that only policy rule data is being collected, 25 policy rules exist, and the refresh interval is 120.

- Size of log file in bytes (409600) * number of log files (5) / 232 = number of records (8827)
- Number of records (8827) / number of policies (25) = number of refresh cycles (353)
- Number of refresh cycles (353) * refresh interval in minutes (2) = 706 minutes worth of data

The previous formulas can be reversed to help arrive at the needed size of the log files:

- Number of refresh cycles = minutes worth of data / refresh interval in minutes
- Number of records = number of refresh cycles * number of policies
- Size of log file in bytes = (number of records * 232) / number of log files

PolicyServer statement

The Policy Agent acting as a policy client uses the PolicyServer statement to determine what type of policies to retrieve from the policy server. This statement also specifies security information and processing information that is passed to the policy server.

Requirement: Connectivity to the policy server is needed for all images that specify the PolicyServer statement.

Restriction: The PolicyServer statement can appear only in an image configuration file (unless the main and image configuration files are the same file).

Results:

- If a ServerConnection statement is not configured in the main configuration file, then this statement is ignored.
- For a policy type, if remote policies are used, then the local policies of the same type are ignored.
- The policy client disconnects from the policy server when one of the following occurs:
  - The ServerConnection or PolicyServer statement is removed. The result is that all remote policies are uninstalled. If local policies are configured, then they are installed.
  - The PolicyServer statement is updated and all PolicyType parameters are removed. The result is that the remote policies for the associated TCP/IP stack are uninstalled. If the local policies for the associated TCP/IP stack are configured, they are installed.
- The policy client disconnects from and reconnects to the policy server when one of the following occurs:
  - The PolicyServer statement is updated and the client name, user identification or authorization parameters have changed.
  - The connection between the policy server and the policy client ends.

- If a PolicyType parameter is removed, then the remote policies for this policy type are removed for the associated TCP/IP stack. If the local policies for this policy type are configured, they are installed.

You must have defined security product authority in the SERVAUTH class for the policy client's user ID. The policy client's client name is used as the image name. For details, see the general policy agent configuration information in z/OS Communications Server: IP Configuration Guide. Wildcard values are allowed in profile names. The following example shows the structure of the security product profile:

```
EZB.PAGENT.sysname.image.ptype
```

If a PolicyServer statement appears multiple times in an image configuration file, the last occurrence of the statement is used. If the PolicyServer statement appears in the main configuration file, it is ignored (unless the main and image configuration files are the same file).

Syntax

```
►►──PolicyServer──┤ Place Braces and Parameters on Separate Lines ├──────────────►◄
```

Place Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬──────────────────────────────────────────┤
   ├─┤ PolicyServer Parameters ├──┤
   └─}────────────────────────────┘
```

PolicyServer Parameters:

```
├──Userid userid──AuthBy──┬─Password password─┬──┬───────────────────────┬──────►
                          └─Passticket────────┘  └─ClientName clientName─┘


    ┌──────────────────────────────────────────────────────┐
    ▼                                                      │
►──┬────────────────────────────────────────────────────────┬─────────────────┤
   └─PolicyType──┬─IDS─────┬──┤ PolicyTypeSpecification ├─┘
                 ├─IPSec───┤
                 ├─QoS─────┤
                 ├─Routing─┤
                 └─TTLS────┘
```

PolicyTypeSpecification:

```
├──┬─{────────────────────────────────────────────┬────────────────────────────┤
   ├─┤ PolicyTypeSpecification Parameters ├─────┤
   └─}──────────────────────────────────────────┘
```

PolicyTypeSpecification Parameters:

```
├──┬─────────┬──┬─────────┬──────────────────────────────────────────────────┤
   ├─FLUSH───┤  ├─PURGE───┤
   └─NOFLUSH─┘  └─NOPURGE─┘
```

Parameters

`UserId`

Specifies the policy client's user identification string. The policy server uses this parameter to identify which resources the client can access. The user ID is a string 1 - 8 alphanumeric characters in length. The first character cannot be a number. No special characters, such as the following, are allowed.

at sign (@)

dollar sigh ($)

number sign (#)

asterisk (*)

`AuthBy`

Indicates which method the policy server uses for authentication of the user ID. The options are Password and the more secure PassTicket.

`Password`

This option causes the client to send the configured password to the policy server for authentication. The password is 1 - 8 characters in length.

Rule: The password must match the password defined in the security product for the user ID.

`PassTicket`

The PassTicket option causes the client to generate a one-time session key. See the information about the secured signon function in z/OS Security Server RACF Security Administrator's Guide.

`ClientName`

A string 1 - 24 characters in length that specifies the client name (PEPInstance name) for this policy client. This client name is used by the policy server to determine which configuration files to use to load the client's policies and whether proper security authorization is configured. See "DynamicConfigPolicyLoad statement" on page 961 for details about how this name is used to select the remote configuration file names.

Result: If no client name is configured, then the policy client generates this parameter based on the system's host name and the associated TcpImage or PEPInstance statement image name.

For example, if the system host name is MVSIBM and TcpImage name is TCPCS, then the generated client name is MVSIBM_TCPCS.

`PolicyType`

Indicates what policy types the policy client retrieves from the policy server.

Tip: If the policy client or policy server is running release V1R9, do not define QoS policies with PolicyScope TR if you retrieve QoS and IDS policies differently (one local and the other remote). These types of policies are transformed into IDS policies by the Policy Agent; there are different implications depending on which of the following policy types are locally or remotely retrieved:

- When IDS policies are retrieved locally and QoS policies are retrieved remotely, the following outcomes result:
  - Any TR scope policies that are specified in local QoS configuration files do not exist because the local QoS configuration files are not read.
  - Any TR scope policies that are specified in remote QoS configuration files do not exist because they exist as IDS policies on the policy server; IDS policies are retrieved locally.
- When IDS policies are retrieved remotely and QoS policies are retrieved locally, any local TR scope policies do not exist, because the are not part of the remote IDS configuration. The local TR scope policies are installed for the interval of time between when the local configuration is read and the remote configuration is retrieved. This interval of time might be lengthy if the policy server cannot be contacted.

Results:

- When QoS policies are retrieved remotely, any QoS policies retrieved from the LDAP server are discarded. When IDS policies are retrieved remotely, any IDS policies retrieved from the LDAP server are discarded.
- If you retrieve both QoS and IDS policies remotely, the ReadFromDirectory statement is ignored. This means no QoS or IDS policies being retrieved from the LDAP server.
- If you specify the policy type IPSec, but IPSec policies are not enabled for the client's TCP/IP stack, then this parameter is ignored. This means that no IPSec policies are retrieved from the policy server. To enable IPSec in the TCP/IP stack, use the IPSECURITY parameter on the IPCONFIG statement in the TCP/IP profile.

FLUSH

Specifies that all policies installed in the Policy Agent and the TCP/IP stack are deleted at the following times:

- When a new TcpImage statement is processed for the first time, including starting Policy Agent.
- When a MODIFY PAGENT,REFRESH command is entered.

NOFLUSH

Specifies that all policies installed in the Policy Agent and the TCP/IP stack are to remain during initial startup and at each refresh interval. In addition, policies that are deleted from a configuration are not deleted from the Policy Agent or the TCP/IP stack.

PURGE

Specifies that all policies installed in the TCP/IP stack are deleted during normal termination, and also when a TcpImage or PEPInstance statement is deleted.

NOPURGE

Specifies that all policies installed in the TCP/IP stack remain during normal termination and when a TcpImage or PEPInstance statement is deleted.

For details, see the FLUSH and PURGE information in z/OS Communications Server: IP Configuration Guide.

Results:

- The FLUSH, NOFLUSH, PURGE, and NOPURGE parameters are ignored for the policy types IPSec and Routing.

- To delete all remote policies for a given policy type, delete the appropriate PolicyType parameter from the PolicyServer statement. This deletes these policies from the policy client and the policy server.

QOSConfig statement

Use the QOSConfig statement to specify the path of a local QoS policy file that contains stack-specific QoS policy statements.

Results: For the associated TCP/IP image on the policy client, if the PolicyServer statement specifies remote QoS policies, then one the following situations occurs:
- If no local QoS policies are installed, then the QOSConfig statement is ignored.
- If local QoS policies are already installed, the result is the same as if the QoSConfig statement had been deleted.

The refresh interval for the QOSConfig file is inherited from the image configuration file containing the corresponding QOSConfig statement.

The QOSConfig statement can appear only in an image configuration file. If a QOSConfig statement appears multiple times in an image configuration file, the last occurrence of the statement is used. If the QOSConfig statement appears in the main configuration file, it is ignored (unless the main and image configuration files are the same file).

Syntax

```
►►──QOSConfig──path──────────────────────────────────────────────────►◄
```

Parameters

path
The path of the stack-specific QOS policy file to be installed.

You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

```
QOSConfig  //'USER1.PAGENT.CONF(QOS)'
QOSConfig  /u/user1/pagent.qos
```

Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported only for z/OS UNIX files; MVS data sets are not monitored for change.

Results:
- If the QOSConfig statement is not specified, then QoS policies are defined in the image configuration file.
- If the QOSConfig statement is deleted and FLUSH is configured, then all QoS policies that are defined in this QoS policy file are deleted from the corresponding stack.

ReadFromDirectory statement

Use the ReadFromDirectory statement to initialize Policy Agent as an LDAP client. The policies are downloaded from the LDAP server, along with the policies specified in this Policy Agent configuration file (the current one being used by Policy Agent that contains this statement). All the policies are installed to the appropriate TCP images.

You can use a set of sample files to help set up the LDAP server and populate it with policies. These files reside in the /usr/lpp/tcpip/samples directory.

One set of sample files defines the schema object classes and attributes for LDAP protocol version 3 servers. These files are:
- pagent_r8qosschema.ldif
- pagent_r5idsschema.ldif

Requirement: These files must be installed on the LDAP server as a subschema of the cn=schema object by using the command.

See the prologs in these sample files and z/OS Communications Server: IP Configuration Guide for more information.

The remaining sample files are examples of policy objects that can be installed on an LDAP server after the schema has been defined using this schema definition files. These files are:
- pagent.ldif contains a top level structure of policy objects.
- pagent_starter_IDS.ldif contains a starter set of IDS policies.
- pagent_starter_QoS.ldif contains a starter set of QoS policies.
- pagent_advanced_IDS.ldif contains an advanced set of IDS policies.
- pagent_advanced_QoS.ldif contains an advanced set of QoS policies.

See the prologs in these sample files and z/OS Communications Server: IP Configuration Guide for more information.

Tip: These policies are not intended to be used as shipped, but they can be copied to a *custom* set (defined in pagent.ldif) and modified as needed.

For more information about how to use LDAP and for other LDAP references, see *Understanding LDAP (SG24–4986)*.

Syntax

```
►►──ReadFromDirectory──┤ Place Braces and Parameters on Separate Lines ├──────────►◄
```

Place Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────────┬──────────────────────────────────────┤
   ├─┤ ReadFromDirectory Parameters ├──┤
   └─}─────────────────────────────────┘
```

ReadFromDirectory Parameters:

```
   ┌─LDAP_Server ──127.0.0.1─┐  ┌─LDAP_Port ──389──┐
├──┴─LDAP_Server ──address───┴──┴─LDAP_Port ──port─┴──────────────────────────►
```

```
                                            ┌─LDAP_BackupPort ──389──┐
►─┬────────────────────────────────┬───────┼────────────────────────┼──────────────►
  └─LDAP_BackupServer ──address──┘        └─LDAP_BackupPort ──port─┘

►─┬──────────────────────────────────────────────────────────────┬────────────────►
  └─LDAP_DistinguishedName ──string──LDAP_Password ──string──┘

  ┌─LDAP_SessionPersistent ──No──────┐  ┌─LDAP_ProtocolVersion ──3──┐
►─┼──────────────────────────────────┼──┼───────────────────────────┼──────────────►
  └─LDAP_SessionPersistent──┬─Yes─┬─┘  └─LDAP_ProtocolVersion────3─┘
                             └─No──┘

  ┌─LDAP_SchemaVersion ──3────┐
►─┼───────────────────────────┼──┬────────────────────┬────────────────────────────►
  └─LDAP_SchemaVersion──┬─1─┤   └─Base ──string─┘
                         ├─2─┤
                         └─3─┘

►─┬──────────────────────────────┬──┬──────────────────────────────────┬──────────►
  └─LDAP_SelectedTag ──string──┘   └─SearchPolicyBaseDN ──string─┘

►─┬────────────────────────────────────────┬──┬──────────────────────────┬────────►
  │ ┌──────────────────────────────────┐ │  │ ┌─────────────────────┐ │
  │ ▼                                    │ │  │ ▼                      │ │
  └───SearchPolicyKeyword ──keyword──┴─┘  └───PolicyRole ──role──┴─┘

►─┬────────────────────────────────────────────────┬──────────────────────────────►
  │ ┌────────────────────────────────────────┐ │
  │ ▼                                          │ │
  └───SearchPolicyGroupKeyWord ──string──┴─┘

                                                ┌─LDAP_AbstractPolicy ──Yes──────┐
►─┬─────────────────────────────────────────┬──┼────────────────────────────────┼─►
  │ ┌───────────────────────────────────┐ │  │                   ┌─Yes─┐      │
  │ ▼                                     │ │  └─LDAP_AbstractPolicy──┴─No──┴──────┘
  └───SearchPolicyRuleKeyWord ──string──┴─┘

►─LDAP_SSL──┤ Place Braces and Parameters on Separate Lines ├──────────────────────┤
```

Place Braces and Parameters on Separate Lines:

```
├──┬─{───────────────────────────┬───────────────────────────────────────────────┤
   ├─┤ LDAP_SSL Parameters ├─┤
   └─}───────────────────────────┘
```

LDAP_SSL Parameters:

```
├──LDAP_SSLKeyringFile ──filename──┬──────────────────────────────────────────┬───►
                                      └─LDAP_SSLKeyringPassword ──password──┘
```

```
►─┬──────────────────────┬────────────────────────────────────────────────────────┤
  └─LDAP_SSLName ──string─┘
```

Parameters

LDAP_Server
: The name of the server that contains policy definitions. The name can be specified as a character string (for example, 'ldapserver.mynetwork.com') or as an IPv4 address (for example, 9.11.12.13). The default is the LDAP server in the local host (127.0.0.1).

LDAP_Port
: The port on which the directory server is running. If not specified, the default, well-known LDAP port of 389, is used.

LDAP_BackupServer
: This attribute specifies the name or IPv4 address of the backup LDAP server for which the search is performed if the Policy Agent cannot connect to the LDAP server as specified in the LDAP_Server and LDAP_Port parameters. The default is no backup server.

LDAP_BackupPort
: This attribute specifies the port number on which the backup LDAP server is running. The default is the well-known LDAP port 389.

LDAP_DistinguishedName
: This attribute is a character string value that specifies the distinguished name for user ID to connect to the LDAP server. If this attribute is not specified, anonymous user ID is used for the connect. If this attribute is specified, LDAP_Password must also be specified.

  Restriction: Case sensitivity of this attribute is determined by the LDAP server.

LDAP_Password
: The password of the connection to the LDAP server. If this attribute is specified, LDAP_DistinguishedName must also be specified.

LDAP_SessionPersistent
: A string that specifies whether the LDAP session with the directory server should be kept open or closed during an update interval time. If this value is not specified, the session is closed after every query from the directory server. Valid values are yes or no. If the LDAP session update interval is small, the value of keeping the session open is greater, because it reduces the overhead of opening the session for each query.

LDAP_ProtocolVersion
: This attribute indicates to Policy Agent what version of the LDAP protocol to use. The default value is 3.

LDAP_SchemaVersion
: This attribute indicates to Policy Agent what version of the schemas to retrieve from LDAP. The value can be 1, 2, or 3. The value should be selected based on your LDAP configuration. The default value is 3.

Base
: The distinguished name of the subtree in the directory containing the policies.

  Requirement: This is required when using schema Version 1 only.

LDAP_SelectedTag
: A string used to select a subset of the policies under the base tree. If not specified, the first machine name returned by gethostname is used.

Restriction: This is allowed only when using schema Version 1.

SearchPolicyBaseDN
This attribute is a character string value (a base distinguished name) that is used as a key to search the LDAP server for policies. It is considered as the initial subtree/group/object to start the search.

Requirement: This attribute is only allowed, and is required, if LDAP_SchemaVersion 2 or higher is specified.

Guideline: Case-sensitivity of this attribute is determined by the LDAP server.

SearchPolicyKeyword
This attribute specifies a generic search keyword to match against all policy objects. Use this attribute to filter the policy objects to be retrieved.

Restriction: This attribute is valid only with LDAP_SchemaVersion 3. You can specify up to eight instances of this attribute. Specify either a single keyword delimited by blanks or any string containing blanks or other special characters, contained in double quotation marks. For example:

```
SearchPolicyKeyword     singleword
SearchPolicyKeyword     "quoted string"
```

SearchPolicyGroupKeyWord
This attribute is a character string value used to scope the search for all group objects.

Restrictions:

- Only policy groups that have a matching PolicyGroupKeywords attribute are returned in the initial search.
- This attribute is allowed only if LDAP_SchemaVersion 2 or higher is specified.

This is similar to the LDAPSelectedTag attribute that is used with LDAP_SchemaVersion 1.

Guidelines:

- Up to eight instances of this attribute are allowed.
- Case-sensitivity of this attribute is determined by the LDAP server.

SearchPolicyRuleKeyWord
This attribute is a character string value that allows users to limit the scope of the policyRule search.

Restrictions:

- Only policy rules that have a matching policyRuleKeywords attribute are returned in the initial search.
- This attribute is allowed only if LDAP_SchemaVersion 2 or higher is specified.

This attribute can also be used when there is no group association in the LDAP server (for example, there is no group hierarchy defined, only rule objects exist) for the policyRule objects.

Guidelines:

- Up to eight instances of this attribute are allowed.
- Case-sensitivity of this attribute is determined by the LDAP server.

PolicyRole
Specifies a policy role or role-combination. Use this parameter to filter the policy rules to be retrieved.

Restriction: This parameter is valid only with LDAP_SchemaVersion 3.

Guidelines:

- This parameter can be repeated as many times as necessary.
- Either a single role or a set of roles, known as a role-combination, can be specified.
- The roles can be single words, or any strings containing blanks or other special characters, contained in double quotation marks.

Role-combinations are specified as follows. The first role is specified the same way that a single role is specified. Each additional role in the role-combination is prefixed with the characters &&. For example:

```
PolicyRole     role1
PolicyRole     &&"quoted role 2"
PolicyRole     "quoted role 3"
PolicyRole     role4
```

Use this parameter to filter out policy rules that do not contain any of the specified roles or role-combinations, using the attribute ibm-policyRoles. For example, the set of roles specified in this example result in the retrieval of any policy rules that specify "role1&&quoted rule 2" or "quoted role3" or "role4" in their ibm-policyRoles values.

LDAP_AbstractPolicy
Specifies whether or not the Policy Agent should search the LDAP server using a search filter that only selects *policy* object classes. Valid values are YES or NO, and YES is the default. If the LDAP server supports matching of auxiliary classes for the objectClass attribute, specify YES. Otherwise, specify NO. This attribute is valid only with LDAP_SchemaVersion 3 and LDAP protocol version 3.

LDAP_SSL
Indicates that additional SSL parameters follow.

LDAP_SSLKeyringFile
LDAP_SSLKeyringFile is the name of the key ring file created by gskkyman. It usually contains the certificates of the trusted (by the client) Certificate Authorities. It can also contain a public key and the associated certificate.

Restriction: This is only needed when client authentication is required.This attribute is required when LDAP_SSL is specified.

LDAP_SSLKeyringPassword
LDAP_SSLKeyringPassword is the password which protects the key ring file. It is set when the key ring file is created with the gskkyman tool.

LDAP_SSLName
LDAP_SSLName is a case-sensitive value that specifies the label assigned when creating a private key/certificate pair with gskkyman. This is used when the client is authenticated.

Restriction: Some servers do not support client authentication; therefore, this parameter is not used.

Examples

The following is a Version 1 schema example:

```
ReadFromDirectory
        {
             Ldap_server     ldapserver.mynetwork.com
```

```
          Ldap_port       9000
          Base            o=ibm,c=us
          Ldap_selectedtag      MVS1
     }
```

The following is a Version 2 schema example:

```
ReadFromDirectory
        {
             LDAP_Server 9.11.12.13
             LDAP_Port 9000
             LDAP_SessionPersistent Yes
             LDAP_BackupServer 9.11.22.23
             LDAP_BackupPort 555
             LDAP_DistinguishedName cn=root, o=IBM, c=US
             LDAP_Password secret
             LDAP_SchemaVersion 2
             LDAP_ProtocolVersion 3
             SearchPolicyBaseDN o=ibm, c=us
             SearchPolicyGroupKeyword MVSA
             SearchPolicyRuleKeyword cherryPicker
             SearchPolicyRuleKeyword ripe
        }
```

The following is a Version 3 schema example:

```
ReadFromDirectory
        {
             LDAP_Server ldapv3server
             LDAP_BackupServer 10.100.1.5
             LDAP_BackupPort 7500
             LDAP_DistinguishedName cn=root, o=IBM, c=US
             LDAP_Password secret
             LDAP_SchemaVersion 3
             LDAP_ProtocolVersion 3
             LDAP_AbstractPolicy Yes
             SearchPolicyBaseDN cn=policy, o=ibm, c=us
             SearchPolicyKeyword QoS
             SearchPolicyKeyword Diffserv
       }
```

RoutingConfig statement

Use the RoutingConfig statement to specify the path of a local Routing policy file that contains stack-specific Routing policy statements.

Requirement: The RoutingConfig statement is required to define Routing policy for a given stack.

Result: For the associated TCP/IP image on the policy client, if the PolicyServer statement specifies remote Routing policies, then the following occurs:

- If no local Routing policies are installed, then the RoutingConfig statement is ignored.
- If local Routing policies are already installed, then the result is the same as if the RoutingConfig statement had been deleted.

Specify the RoutingConfig statement without a path name in each image configuration file to define a common set of policies for multiple stacks.

The refresh interval for the RoutingConfig file is inherited from the image configuration file containing the corresponding RoutingConfig statement.

Restriction: The RoutingConfig statement can appear only in an image configuration file.

If a RoutingConfig statement appears multiple times in an image configuration file, the last occurrence of the statement is used. If the RoutingConfig statement appears in the main configuration file, it is ignored (unless the main and image configuration files are the same file).

Syntax

```
►►──RoutingConfig──┬──────┬──────────────────────────────────────────────►◄
                   └─path─┘
```

Parameters

path
: Specifies the path of the stack-specific Routing policy file to be installed. If no path name is specified, the common Routing policy file specified on the CommonRoutingConfig statement is used.

  You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

  ```
  RoutingConfig  //'USER1.PAGENT.CONF(ROUTING)'
  RoutingConfig  /u/user1/pagent.routing
  ```

  Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported only for z/OS UNIX files; MVS data sets are not monitored for changes.

  Results:

  - After a TCP/IP stack has been recycled, all active policies are reinstalled.
  - If the RoutingConfig statement is deleted, all Routing policies are deleted from the corresponding stack.

ServerConnection statement

The Policy Agent acting as a policy client uses the ServerConnection statement to connect to the Policy Agent acting as a policy server. This statement includes security information and the location of the policy server. The policy client uses this connection to retrieve remote policies. See "PolicyServer statement" on page 977 for more details.

Results:

- An error is flagged if both the ClientConnection and ServerConnection statements are configured on the same Policy Agent. As a result, there is no connection between the policy server and policy client.
- If a PolicyServer statement is not configured in any image configuration file, this statement is ignored, and no connections to the policy server exist.
- If any parameters on the ServerConnection statement are updated after a remote connection is established, the changed values take effect for new connections. The ServerConnectWait and ServerConnectRetries parameters take effect immediately for any connections that require retry processing.

- If the ServerConnection statement is deleted, all established remote connections are stopped. As a result, all remote policies are uninstalled. If configured, then the local policies are now installed.

Syntax

```
►►──ServerConnection──┤ Place Braces and Parameters on Separate Lines ├──────────►◄
```

Place Braces and Parameters on Separate Lines:

```
├──┬─{────────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ ServerConnection Parameters ├─┤
   └─}────────────────────────────┘
```

ServerConnection Parameters:

```
                    ┌─ServerPort 16310─┐
├──ServerHost host──┼──────────────────┼──┬─────────────────────────┬────────────►
                    └─ServerPort port──┘  └─ServerHostBackup host───┘

   ┌─ServerPortBackup 16310─┐  ┌─ServerConnectWait 60──────┐
►──┼────────────────────────┼──┼───────────────────────────┼─────────────────────►
   └─ServerPortBackup port──┘  └─ServerConnectWait seconds─┘

   ┌─ServerConnectRetries 3─────┐
►──┼────────────────────────────┼──┬───────────────────────────────────────┬─────┤
   └─ServerConnectRetries times─┘  └─ServerSSL | ServerSSL Specification───┘
```

ServerSSL Specification:

```
├──┬─{──────────────────────┬────────────────────────────────────────────────────┤
   ├─┤ ServerSSL Parameters ├─┤
   └─}──────────────────────┘
```

ServerSSL Parameters:

```
├──┬───────────────────────────┬──┬─────────────────────────────────────────┬────►
   └─ServerSSLKeyring value────┘  └─ServerSSLKeyringPassword password───────┘

►──┬─────────────────────────────────────────┬──┬──────────────────────────┬─────►
   └─ServerSSLKeyringStashFile filename──────┘  └─ServerSSLName string─────┘

                                        ┌─ServerSSLv3 Off─┐
►──┬──────────────────────────────────┬─┼─────────────────┼──────────────────────┤
   └─ServerSSLV3CipherSuites ciphers──┘ └─ServerSSLv3 On──┘
```

Parameters

`ServerHost`
A string 1 - 512 characters in length that specifies the name of the primary policy server host that contains policy definitions. Specify the name as a host

name (for example, policyserver.mynetwork.com) or as an IP address (for example, 9.11.12.13). The IP address can be either IPv4 or IPv6. This parameter is required.

ServerPort
The policy server listening port number. The policy client connects using this port number.

The valid port values are in the range 1 - 65 535. The default is 16 310.

ServerHostBackup
A string 1 - 512 characters in length specifying the name of the backup policy server that contains policy definitions. The name can be specified as a host name (for example, policyserver.mynetwork.com) or as an IP address (for example, 9.11.12.13). The IP address can be either IPv4 or IPv6. The default is no backup server.

ServerPortBackup
The backup policy server listening port number. The policy client connects using this port number.

The valid port values are in the range 1 - 65 535. The default is 16 310.

Result: This parameter is ignored if the ServerHostBackup parameter is not specified.

ServerConnectWait
Specifies the number of seconds (1-300) that the policy client waits between connection attempts when trying to establish a connection with a policy server. The default value is 60 seconds.

The product of the ServerConnectWait value multiplied by the ServerConnectRetries value defines the maximum number of seconds that the policy client attempts to connect with a policy server before switching to another policy server (if a backup server is configured). For example, if the ServerConnectWait value is 60 and the ServerConnectRetries value is 3, then the policy client waits a maximum of 180 seconds for a successful connection.

This parameter is also used if policies cannot be loaded from the policy server. The policy client waits for the specified amount of time before trying to load the policies again.

ServerConnectRetries
Specifies the number of times (1-10) that the policy client attempts to establish a connection with a policy server. The default value is 3 retries.

The product of the ServerConnectWait value multiplied by the ServerConnectRetries value defines the maximum number of seconds that the policy client attempts to connect with a policy server before switching to another policy server (if a backup server is configured) . For example, if the ServerConnectWait value is 60 and the ServerConnectRetries value is 3, then the policy client waits a maximum of 180 seconds for a successful connection.

ServerSSL
Indicates that additional SSL parameters follow.

This parameter is optional. If you want to use a secure connection to the policy server, specify this parameter and other SSL parameters as needed.

ServerSSLKeyring
A string 1 - 1023 characters in length specifying the name of the key ring file created by gskkyman, or the ring name of the SAF key ring. This key ring usually contains the certificates of the trusted (by the client) Certificate

Authorities. The key ring can also contain a public key and the associated certificate (this is needed only when client authentication is required).

This parameter is required if ServerSSL is specified.

ServerSSLKeyringStashFile
A string 1 - 1023 characters in length that specifies the name of the key ring stash file. The stash file is created when the key ring file is created with the gskkyman tool. If the ServerSSLKeyringPassword parameter is specified, then it is used instead of this parameter. The password and the stash file parameter are not required with a SAF keyring.

ServerSSLKeyringPassword
A string 1 - 128 characters in length that specifies the password for the key database that protects the key ring file. It is set when the key ring file is created with the gskkyman tool. This parameter is optional. As an alternative, you can specify the more secure ServerSSLKeyringStashFile parameter to use a key database that was created with the gskkyman tool. The password and the stash file parameter are not required with a SAF key ring.

ServerSSLName
A string 1 - 256 characters in length specifying a case-sensitive value that specifies the label assigned when creating a private key/certificate pair with gskkyman. This value is used when the client is authenticated.

Rules (for servers that use client authentication:

- If the AT-TLS policy on the policy server specifies HandshakeRole Server, the ServerSSLName parameter must specify the name of the server's certificate.
- If the AT-TLS policy on the policy server specifies HandshakeRole ServerWithClientAuth, the ServerSSLName parameter must specify the name of the client's certificate.

Restriction: Some servers do not support client authentication; therefore, this parameter is not used.

ServerSSLV3CipherSuites

| **Tip:** If the System SSL needs to access z/OS Integrated Cryptographic Services
| Facility (ICSF), ICSF must be started before you start the Policy Agent. For
| more information about using hardware Cryptographic Features with System
| SSL, see z/OS Cryptographic Services System SSL Programming.
| Specifies the SSL version 3 TLS version 1.0, TLS version 1.1, or TLS version 1.2
| cipher suites in order of preference. If a ServerSSLV3CipherSuites parameter is specified more than once, the values are concatenated to create a single list of cipher suites. For System SSL, the GSK_V3_CIPHER_SPECS value is set to the concatenated value. The *ciphers* value is a string of one or more 2-hexadecimal
| character ciphers that are SSL version 3 TLS version 1.0, TLS version 1.1, or
| TLS version 1.2 ciphers, or a single cipher constant. The cipher string cannot have blanks between each cipher. If duplicate ciphers, the first instance of the cipher is used and all other instances you specify are ignored. The maximum number of ciphers is 255. For System SSL, see the description of the gsk_environment_open() call in z/OS Cryptographic Services System SSL Programming for a list of valid cipher suites. Table 68 on page 992 lists cipher constants that are supported.

Table 68. Supported cipher constants for the ServerSSLV3CipherSuites parameter

Cipher constant	Hexadecimal character
TLS_NULL_WITH_NULL_NULL	00
TLS_RSA_WITH_NULL_MD5	01
TLS_RSA_WITH_NULL_SHA	02
TLS_RSA_EXPORT_WITH_RC4_40_MD5	03
TLS_RSA_WITH_RC4_128_MD5	04
TLS_RSA_WITH_RC4_128_SHA	05
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	06
TLS_RSA_WITH_DES_CBC_SHA	09
TLS_RSA_WITH_3DES_EDE_CBC_SHA	0A
TLS_DH_DSS_WITH_DES_CBC_SHA	0C
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	0D
TLS_DH_RSA_WITH_DES_CBC_SHA	0F
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	10
TLS_DHE_DSS_WITH_DES_CBC_SHA	12
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	13
TLS_DHE_RSA_WITH_DES_CBC_SHA	15
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	16
TLS_RSA_WITH_AES_128_CBC_SHA	2F
TLS_DH_DSS_WITH_AES_128_CBC_SHA	30
TLS_DH_RSA_WITH_AES_128_CBC_SHA	31
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	32
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	33
TLS_RSA_WITH_AES_256_CBC_SHA	35
TLS_DH_DSS_WITH_AES_256_CBC_SHA	36
TLS_DH_RSA_WITH_AES_256_CBC_SHA	37
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	38
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	39
TLS_RSA_WITH_NULL_SHA256	3B
TLS_RSA_WITH_AES_128_CBC_SHA256	3C
TLS_RSA_WITH_AES_256_CBC_SHA256	3D
TLS_DH_DSS_WITH_AES_128_CBC_SHA256	3E
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	3F
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	40
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	67
TLS_DH_DSS_WITH_AES_256_CBC_SHA256	68
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	69
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	6A
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	6B
TLS_RSA_WITH_AES_128_GCM_SHA256	9C
TLS_RSA_WITH_AES_256_GCM_SHA384	9D

Table 68. Supported cipher constants for the ServerSSLV3CipherSuites parameter (continued)

Cipher constant	Hexadecimal character
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	9E
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	9F
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	A0
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	A1
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	A2
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	A3
TLS_DH_DSS_WITH_AES_128_GCM_SHA256	A4
TLS_DH_DSS_WITH_AES_256_GCM_SHA384	A5

ServerSSLv3
Indicates whether SSLv3 is enabled for the policy client that connects to the server.

On	SSLv3 is enabled.
Off	SSLv3 is disabled. This is the default.

ServicesConnection statement

The Policy Agent uses the ServicesConnection statement to specify the listening port, listening TCP/IP image, and security level for connections to the Policy Agent. Applications that use this connection are known as services requestors. One such services requestor is the IBM Configuration Assistant for z/OS Communications Server, which is an import requestor that uses this connection to retrieve import policies or TCP/IP profile information.

Consider the following characteristics when using the ServicesConnection statement:

- If you want to use default values for all parameters, you can specify the ServicesConnection statement without a set of braces.
- If you specify Security Basic, you can either use a default unsecure connection or supply user defined AT-TLS policies for this import services connection to create a secure SSL connection.
- If you specify Security Secure, the Policy Agent generates an AT-TLS policy and installs it at the lowest priority (lower than any configured policies) into the specified TCP/IP image after any configured local or remote AT-TLS policies have been installed.
- If you update any parameters that are used in the generated policy (Port, Trace or Keyring parameters), the Policy Agent reinstalls the generated policy.
- The Policy Agent listens for TCP connections on the specified TCP/IP image name only.
- If you remove the ServicesConnection statement, all services requestor connections to this Policy Agent are disconnected.
- Updates to the ServicesConnection statement are used for only new services requestor connections to the Policy Agent.
- If you do not configure the ServicesConnection statement, or the image name is not an active TCP/IP image, the Policy Agent does not listen on any port for services requestor connections.

- To restart the listen for services requestor connections and to reinstall the generated AT-TLS policy, issue the MODIFY SRVLSTN command. See z/OS Communications Server: IP System Administrator's Commands for more information about this command.

Syntax

```
►►──ServicesConnection──┤ Place Braces and Parameters on Separate Lines ├────────►◄
```

Place Braces and Parameters on Separate Lines:

```
├──┬──────────────────────────────────────┬─────────────────────────────────────┤
   ├─{────────────────────────────────────┤
   ├─┤ ServicesConnection Parameters ├────┤
   └─}────────────────────────────────────┘
```

ServicesConnection Parameters:

```
   ┌─Port 16311─┐                                 ┌─Security Basic──────────┐
├──┼────────────┼──┬─────────────────────────┬──┼─────────────────────────┼──────►
   └─Port value─┘  └─ImageName imagename─┘      └─Security──┬─Secure─┬──┘
                                                            └─Basic──┘

   ┌─Trace 2─┐
►──┼─────────┼──┬─────────────────┬────────────────────────────────────────────┤
   └─Trace n─┘  └─Keyring value─┘
```

Parameters

Port
: Specifies the port that the Policy Agent listens on for TCP connections from services requestors on the specified TCP/IP image name. If you are using the IBM Configuration Assistant for z/OS Communications Server, this port must be the same as the host connection port that is specified on the Configuration Assistant Import Policy Data request panel for any import requestor that connects to this Policy Agent or on the request panels for discovery import (for example, on the Discover Stack Local Addresses panel).

  If you change the Port parameter value, the Policy Agent listens for new TCP connections using the updated value on the specified TCP/IP image name.

  Valid port values are in the range 1 - 65 535. The default port value is 16 311.

  Restriction: The port value cannot match the port value configured on the ClientConnection statement.

ImageName
: A string 1 - 8 characters in length that specifies the TCP/IP image name. The Policy Agent listens for services connections only on this TCP/IP image.

  If you change the ImageName value, the Policy Agent listens for new TCP connections on the newly specified TCP/IP image name. If you specify the Security parameter with the Secure value and update the ImageName parameter, the Policy Agent removes the generated policy from the original TCP/IP image and installs it on the newly specified image.

If you specify Security Basic and define AT-TLS policies for this import services connection to create a secure SSL connection, these policies must be installed for this ImageName.

Results:

- In a single stack (INET) environment, the Policy Agent uses the active TCP/IP image to listen for services connection requests.
- In a common INET (CINET) environment, if you do not specify the TCP/IP image name, the Policy Agent uses the default TCP/IP image (resolver supplied TCPIPuserid statement or TCPIPjobname statement). If the Policy Agent cannot determine the default TCP/IP image, the Policy Agent uses the name INET.
- If you specify an image name that does not have a corresponding TcpImage or PEPInstance statement, the Policy Agent creates an internal TcpImage statement with default values to represent the specified TCP/IP image. You can specify only 7 (instead of 8) TcpImage or PEPInstance statements.
- If you specify an image name that is not active, the Policy Agent does not listen for services requestor connections until the TCP/IP image becomes active.

`Security`

Indicates the level of security that is used for the services requestor connection. If you change the Security parameter from Secure to Basic, the Policy Agent uninstalls the generated AT-TLS policy from the specified TCP/IP image.

Basic Specifies one of the following connections:

- The connection does not use SSL and is unsecure.
- You define AT-TLS policies for this import services connection to create a secure SSL connection.

Result: If you specify the Security Basic setting without defining AT-TLS policies, the user ID and password that the services requestor provides flow without encryption.

Tip: For secure SSL, it is recommended to configure Security Basic and to supply user defined AT-TLS policies to protect the import service connection with the required SSL/TLS protection.

Secure

Specifies that the connection uses SSL. The Policy Agent installs a generated AT-TLS policy similar to the following example into the specified TCP/IP image to protect the connection.

Restriction: This option supports only TLSv1.0 and is not recommended for secure SSL.

```
TTLSRule                     TTLS_RULE_______________GENERATED
{
   LocalPortRange             <ServicesConnection port value>
   Direction                  Inbound
   TTLSGroupActionRef         TTLS_GROUP_ACTION_______GENERATED
   TTLSEnvironmentActionRef   TTLS_ENVIRONMENT_ACTION_GENERATED
}
TTLSGroupAction   TTLS_GROUP_ACTION_______GENERATED
{
  TTLSEnabled     On
  Trace           <ServicesConnection trace value>
}
TTLSEnvironmentAction  TTLS_ENVIRONMENT_ACTION_GENERATED
{
  HandshakeRole        Server
  TTLSKeyRingParms
  {
    Keyring           <ServicesConnection keyring value>
  }
}
```

Rule: If you specify Security Secure, the Keyring parameter is required.

Trace

Specifies the level of AT-TLS tracing to be used for the generated AT-TLS policy. Valid values for *n* are in the range 0 - 255. The sum of the numbers associated with each level of selected tracing is the value you should specify for *n*. If *n* is an odd number, errors are written to joblog, and all other configured traces are sent to syslogd.

0 No tracing is enabled.

1 (Error)
Errors are traced to the TCP/IP joblog.

2 (Error)
Errors are traced to syslogd. This is the default. The messages are issued with syslogd priority code **err**.

4 (Info)
Enables tracing of instances when a connection is mapped to an AT-TLS rule and when a secure connection is successfully initiated. The messages are issued with syslogd priority code **info**.

8 (Event)
Enables tracing of major events. The messages are issued with syslogd priority code **debug**.

16 (Flow)
Enables tracing of system SSL calls. The messages are issued with syslogd priority code **debug**.

32 (Data)
Enables tracing of encrypted negotiation and headers. This value traces the negotiation of secure sessions. The messages are issued with syslogd priority code **debug**.

64 Reserved.

128 Reserved.

255 Enables all tracing.

If you specify Security Basic, this parameter is ignored.

Keyring

A string 1 - 1 023 in length that specifies the ring name of the SAF key ring. This key ring typically contains the certificates of the trusted (by the client) Certificate Authorities.

Restriction: If Security is configured with Secure, then this parameter is required.

If you specify Security Basic, this parameter is ignored.

SetSubnetPrioTosMask statement

Use the SetSubnetPrioTosMask statement to define a mapping of IPv4 Type of Service (ToS) byte or IPv6 Traffic Class to outbound interface device and virtual

LAN (VLAN) user priority values. It maps priorities for interfaces that use OSA-Express configured in QDIO mode. If this statement is not specified, TCP/IP uses the system default ToS or Traffic Class mask and priority levels for all interfaces currently defined for IPv4 (RFC 791).

The current IPv4 ToS byte format defines the first 3 bits to be the precedence bits (for example, priority). Therefore, the default for the subnet ToS mask, if this statement is not specified, is 11 100 000. The same default mask also applies to IPv6 Traffic Class.

Restrictions:
- Only Queued Direct I/O (QDIO) devices can support priorities.
- A maximum of 16 SetSubnetPrioTosMask statements can be specified for each TCP/IP stack.

This statement sets up ToS or Traffic Class to priority mapping for those devices. QDIO supports four priority levels, 1 - 4, with 4 being the lowest priority. Following is the default mapping of these four priorities to the various ToS byte or Traffic Class values:

TOS	Priority
00000000	4
00100000	4
01000000	3
01100000	2
10000000	1
10100000	1
11000000	1
11100000	1

The ToS byte or Traffic Class is also used by other network devices (for example, routers and switches) to determine the priority of a packet.

Guideline: If Enterprise Extender has set ToS bytes, this overrides those settings.

Tip: An outbound packet with a ToS or Traffic Class value that consists of zeros allows for prioritizing outbound OSA-Express data using the WorkLoad Manager service class importance level. See the information about Prioritizing outbound OSA-Express data using the WorkLoad Manager service class in z/OS Communications Server: IP Configuration Guide for more information about using WorkLoad Manager service class importance level values with OSA-Express QDIO interfaces.

Syntax

```
►►──SetSubnetPrioTosMask──┤ Place Braces and Parameters on Separate Lines ├──────────►◄
```

Place Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────────┬──────────────────────────────────────────┤
   ├─┤ SetSubnetPrioTosMask Parameters ├─┤
   └─}─────────────────────────────────┘
```

SetSubnetPrioTosMask Parameters:

```
   ┌─SubnetAddr ──0.0.0.0─┐
├──┼──────────────────────┼──SubnetTosMask ──mask──────────────────────►
   └─SubnetAddr ──address─┘

   ┌──────────────────────────────────────────────────────┐
   ▼ ┌─PriorityTosMapping 4tos 0──────────────────────────┐ │
►──┴─┼────────────────────────────────────────────────────┼─┴───────────┤
     └─PriorityTosMapping ──priority tos─┬───────────────┬┘
                                         └─user_priority─┘
```

Parameters

SubnetAddr
The local subnet interface address. This can be an IPv4 address or an interface name. A value of 0 indicates that the mask is the same for all interfaces. The default is all interfaces. All interfaces are the same as coding a value of 0, or not specifying this parameter.

Requirement: If an interface name is specified, it must match a name specified on one of the following statements in the TCP/IP profile:

- LINK statement for an IPv4 interface
- INTERFACE statement for an IPv4 or IPv6 interface

SubnetTosMask
SubnetTosMask contains eight bits, left-aligned, for the ToS or Traffic Class mask. For example, 101 would be 10 100 000. There is no default.

Requirement: This is a required parameter.

PriorityTosMapping
Three values to indicate each priority level to ToS or Traffic Class value mapping. The first value of each mapping is an integer to indicate the device priority level, the second value is eight bits, left-aligned, to indicate the ToS or Traffic Class value, and the third value is an optional integer to indicate the user priority (0 - 7, where 0 is the lowest priority). User priority is also known as virtual LAN (VLAN) priority.

Restrictions:

- If this parameter is not specified for a ToS or Traffic Class value, that value maps to a device priority value of 4, and a user priority value of 0.
- A maximum of 32 PriorityTosMapping parameters can be specified.

Result: Coding the virtual LAN (VLAN) user priority causes a frame to be sent out based on the IEEE 802.1Q specification, which establishes a standard method for tagging Ethernet frames with VLAN priority and membership information. Specifically, a VLAN priority-tagged frame is used to convey packet priority to the switches; it has a value of NULL for VLANID. A full VLAN-tagged frame contains both the priority and non-null VLANID. If you have switches in your network that do not support the IEEE 802.1Q standard or that are not properly configured for these types of frames, the frames might be dropped by the switch.

Examples

```
SetSubnetPrioTosMask
{
SubnetAddr NSQDIO3
SubnetTosMask 11100000
PriorityTosMapping 1 11100000 7
PriorityTosMapping 1 11000000 7
PriorityTosMapping 2 10100000 6
PriorityTosMapping 2 10000000 5
PriorityTosMapping 2 01100000 5
PriorityTosMapping 3 01000000 3
PriorityTosMapping 4 00100000 2
PriorityTosMapping 4 00000000 0
}
```

```
SetSubnetPrioTosMask
{
SubnetTosMask 11100000
PriorityTosMapping 1 11100000
PriorityTosMapping 1 11000000
PriorityTosMapping 1 10100000
PriorityTosMapping 1 10000000
PriorityTosMapping 2 01100000
PriorityTosMapping 2 01000000
PriorityTosMapping 3 00100000
PriorityTosMapping 4 00000000
}
```

TcpImage and PEPInstance statement

Use the TcpImage and PEPInstance statements to specify a TCP/IP image and its associated image configuration file to be installed to that image. These statements are synonyms and you can use either of them. If no TcpImage statement is specified, any policy definitions are installed to the default TCP/IP image (resolver supplied TCPIPuserid statement or TCPIPjobname statement). The parameters FLUSH or NOFLUSH can be used to force deletion of some policy types from the stack at startup and certain other events. The parameters PURGE or NOPURGE can be used to delete some policy types from the stack during normal shutdown (for example, KILL or STOP).

A single stack environment is defined in BPXPRMxx parmlib member by setting 'FILESYSTYPE TYPE(INET)'. For more information, see z/OS UNIX System Services Planning.

The Policy Agent uses the TcpImage statement within a single stack environment in the following ways:

- If one or more TcpImage statements are specified, Policy Agent always uses the default TCP/IP image (resolver supplied TCPIPuserid statement or TCPIPjobname statement). All associated policy control statements are installed to the active TCP/IP stack.
- If no TcpImage statement is specified, any control statements are installed to the active TCP/IP stack.
- If Policy Agent cannot determine the name of the TCP/IP image (resolver supplied TCPIPuserid statement or TCPIPjobname statement), INET is the default name used. Any control statements are installed to the active TCP/IP stack.

Result: If the default TCP/IP image (resolver supplied TCPIPuserid statement or TCPIPjobname statement) is not the same as the active TCP/IP stack, the following situations occur:

- The default TCP/IP stack name is used in messages and displays. Policy Agent creates an internal TcpImage statement with default values to represent the specified TCP/IP image.
- The interface used by the stack to inform the Policy Agent about stack restarts does not function.

Syntax

```
                                        ┌─NOFLUSH─┐  ┌─NOPURGE─┐  ┌─1800─┐
►►──┬─TcpImage────┬─name──┬──────┬──┼─────────┼──┼─────────┼──┼──────┼──────────►◄
    └─PEPInstance─┘       └─path─┘  ├─FLUSH───┤  ├─PURGE───┤  └─i────┘
                                    └─NOFLUSH─┘  └─NOPURGE─┘
```

Parameters

name
: The jobname of the TCP/IP image.

  Requirement: The name must be one to eight characters in length.

path
: The path of the image configuration file to be installed. If an image configuration file is not specified, the following policy definitions (if any) in this policy configuration file are installed.

  You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

```
TcpImage  TCPIP1 //'USER1.PAGENT.CONF(TCPIP1)' FLUSH PURGE
TcpImage  TCPIP1 /u/user1/pagent.tcpip1 FLUSH PURGE
```

FLUSH
: Specifies that all policies installed in the Policy Agent and the TCP/IP stack are deleted when:
  - A new TcpImage statement is processed for the first time, including Policy Agent starting.
  - A MODIFY PAGENT,REFRESH command is entered.

NOFLUSH
: Specifies that all policies installed in the Policy Agent and the TCP/IP stack are to remain during initial startup and at each refresh interval. In addition, policies that are deleted from a configuration are not deleted from the Policy Agent or the TCP/IP stack. This is the default.

PURGE
: Specifies that all policies installed in the TCP/IP stack are deleted during normal termination, and also when a TcpImage or PEPInstance statement is deleted.

NOPURGE
: Specifies that all policies installed in the TCP/IP stack remain during normal termination and when a TcpImage or PEPInstance statement is deleted. This is the default.

  For details, see the FLUSH and PURGE considerations in z/OS Communications Server: IP Configuration Guide.

i

An integer that specifies the time interval (in seconds) for checking the creation or modification time of the configuration file or files, and for refreshing policies from the LDAP server. The maximum value is 2 147 483 647. In the following cases, the update interval is changed:

- If a value is not specified, the default is 1 800 seconds (30 minutes).
- If a value of 0 is specified, the default value of 1 800 (30 minutes) is used.
- Any value in the range 1 - 59 is rounded up to 60 seconds (1 minute).

The Policy Agent always uses this time interval to check for changes, but also monitors the configuration file or files in real time if the **-i** startup option is specified. The smallest refresh interval specified on the set of TcpImage statements is used as the refresh interval for the main configuration file.

For example, if **-i** is set to 300, the corresponding configuration files and LDAP server are checked for changes every five minutes. If the configuration file or files have changed within the last 5 minutes, they are read again. The LDAP server (if configured) is also queried again for policies. Any new, changed, or deleted policies are installed to or removed from the stack as appropriate.

Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported only for z/OS UNIX files; MVS data sets are not monitored for changes.

Rules: To dynamically add a TCP/IP stack to the Policy Agent configuration, take one of the following actions in addition to adding the TcpImage statement to the configuration file. This automatically installs active policies.

- If the Policy Agent was started with the **-i** startup option, no further action is necessary. Active policies are automatically installed to the stack when it becomes active.
- If the Policy Agent was not started with the **-i** startup option, take one of the following actions:
  - Issue the MODIFY REFRESH or MODIFY UPDATE command after the stack becomes active. If the MODIFY REFRESH or MODIFY UPDATE command is issued before the stack becomes active, policies are not automatically installed.
  - Wait on the next update interval to check for configuration changes. If the stack is not active, policies are not automatically installed.

Examples

The following example installs the image configuration file */tmp/TCPCS.policy* to the *TCPCS* TCP/IP image, after flushing existing policy control data:

```
TcpImage TCPCS /tmp/TCPCS.policy FLUSH
```

TTLSConfig statement

Use the TTLSConfig statement to specify the path of a local AT-TLS policy file that contains stack-specific AT-TLS policy statements. The TTLSConfig statement is required to define AT-TLS policy for a given stack. To define a common set of policies for multiple stacks, the TTLSConfig statement can be specified without a path name.

Requirement: The TTLSConfig statement is required to define AT-TLS policy for a given stack.

Results: For the associated TCP/IP image on the policy client, if the PolicyServer statement specifies remote AT-TLS policies, then one the following situations occurs:

- If no local AT-TLS policies are installed, then the TTLSConfig statement is ignored.
- If local AT-TLS policies are already installed, then the result is the same as if the TTLSConfig statement had been deleted.

Rule: For AT-TLS policies, if errors are detected during parsing, no new policies are installed.

The FLUSH/NOFLUSH and PURGE/NOPURGE parameters can be used to specify whether or not AT-TLS policies are deleted at startup (and when a MODIFY REFRESH command is entered) and shutdown, respectively.

The refresh interval for the TTLSConfig file is inherited from the image configuration file containing the corresponding TTLSConfig statement.

Specify the TTLSConfig statement without a path name in each image configuration file to define a common set of policies for multiple stacks.

The TTLSConfig statement can appear only in an image configuration file. If a TTLSConfig statement appears multiple times in an image configuration file, the last occurrence of the statement is used. If the TTLSConfig statement appears in the main configuration file, it is ignored (unless the main and image configuration files are the same file).

Syntax

```
►►──TTLSConfig──┬──────┬──┬─────────┬──┬─────────┬────────────────────────────►◄
                └─path─┘  ├─FLUSH───┤  ├─PURGE───┤
                          └─NOFLUSH─┘  └─NOPURGE─┘
```

Parameters

path
: The path of the stack-specific AT-TLS policy file to be installed. If no path name is specified, then the common AT-TLS policy file specified on the CommonTTLSConfig statement is used.

  You can specify an MVS data set name or a z/OS UNIX file name. MVS data set names must be enclosed in single quotation marks (' ') and preceded by a double slash (//). Following are examples of both types of names:

```
TTLSConfig  //'USER1.PAGENT.CONF(TTLS)'
TTLSConfig  /u/user1/pagent.ttls
```

  Restriction: Dynamic monitoring for file updates using the **-i** startup option is supported only for z/OS UNIX files; MVS data sets are not monitored for change.

FLUSH
: FLUSH specifies that all policies installed in the Policy Agent and the TCP/IP stack are deleted. Policies are flushed at the following times:

  - When a new TcpImage statement is processed for the first time, including Policy Agent starting
  - When a MODIFY REFRESH command is entered

NOFLUSH

NOFLUSH specifies that all policies installed in the Policy Agent and the TCP/IP stack are to remain during initial startup and at each refresh interval. In addition, policies that are deleted from a configuration are not deleted from the Policy Agent or the TCP/IP stack.

PURGE

Specifies that all policies installed in the TCP/IP stack are deleted during normal termination, and also when a TcpImage or PEPInstance statement is deleted.

NOPURGE

Specifies that all policies installed in the TCP/IP stack remain during normal termination and when a TcpImage or PEPInstance statement is deleted.

For details, see the FLUSH and PURGE information in z/OS Communications Server: IP Configuration Guide.

Result: If the TTLSConfig statement is deleted and FLUSH configured, then all AT-TLS policies are deleted from the corresponding stack.

AT-TLS policy statements

This topic contains information about the following AT-TLS policy statements:

- "TTLSCipherParms statement" on page 1004
- "TTLSConnectionAction statement" on page 1009
- "TTLSConnectionAdvancedParms statement" on page 1012
- "TTLSEnvironmentAction statement" on page 1017
- "TTLSEnvironmentAdvancedParms statement" on page 1020
- "TTLSGroupAction statement" on page 1028
- "TTLSGroupAdvancedParms statement" on page 1030
- "TTLSGskAdvancedParms statement" on page 1032
- "TTLSGskLdapParms statement" on page 1033
- "TTLSKeyringParms statement" on page 1035
- "TTLSRule statement" on page 1036
- "TTLSSignatureParms statement" on page 1041

Consider the following guidelines when using the AT-TLS policy statements.

Guidelines:

- While configuring AT-TLS policy, see z/OS Cryptographic Services System SSL Programming for a detailed description of each of the System SSL attributes that are being configured using the AT-TLS policy statements (System SSL attributes are those that begin with GSK). See the information describing the gsk_attribute_set_buffer API, the gsk_attribute_set_enum API, and the gsk_attribute_set_numeric_value API descriptions of how each attribute is used by System SSL, as well as the meaning of available attribute settings and default attribute settings.
- AT-TLS requires a valid z/OS UNIX key database, SAF key ring, or z/OS PKCS #11 token. For more information about AT-TLS configuration, see z/OS Communications Server: IP Configuration Guide.

- AT-TLS can be configured to write trace data to syslogd. AT-TLS writes messages to syslogd using the daemon or auth facility. See Chapter 20, "Syslog daemon," on page 909 for more information about configuring syslogd.
- If System SSL needs to access ICSF, ICSF must be started before you start the Policy Agent. For information about using hardware Cryptographic Features with System SSL, see z/OS Cryptographic Services System SSL Programming.

Note the following results when using the AT-TLS policy statements.

Results: When using AT-TLS policy statements, consider the following results:

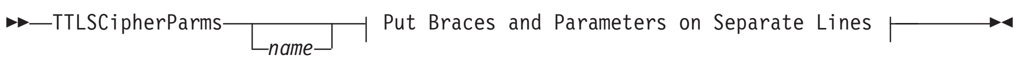
- When an IpAddrGroup statement contains non-continuous ranges of IP addresses, or a PortGroup statement contains non-continuous ranges of port numbers, Policy Agent cannot merge these conditions into a single condition. The group's ranges are displayed by pasearch, as configured, with the summary condition for each of these respective attributes equal to the lowest **from** value in the group to the highest **to** value in the group. If an IP address of value 0.0.0.0 exists in an IpAddrGroup statement, the summary condition for this attribute is set to **All**. If a Port of value 0 exists in a PortGroup statement, the summary condition for this attribute is set to the range **0-0**. When an IpAddrGroup statement contains a mixture of IPv4 and IPv6 addresses, a summary condition cannot be created. The group's ranges are displayed by pasearch, as configured, with a summary condition for this attribute of **All**.
- For optional parameters that have default values and are not specified, pasearch displays the default value when the parameter is not configured.
- For optional parameters that do not have default values and are not specified, pasearch does not display the parameter.
- If an optional parameter is not specified for a GSK statement, System SSL uses its default value.
- For parameters that can be specified in multiple action types, the value used by a connection is determined by the following hierarchical rule set.
  1. If the parameter is specified in the TTLSConnectionAction statement that is the value used.
  2. If the parameter is specified in the TTLSEnvironmentAction statement that is the value used.
  3. If the parameter is specified in the TTLSGroupAction statement that is the value used.
  4. If a default value is defined, that is the value used.
  5. No value is used by AT-TLS and no parameter is explicitly passed to System SSL.
- Each AT-TLS action has a user instance variable (GroupUserInstance, EnvironmentUserInstance, and ConnectionUserInstance). These parameters can be used to cause Policy Agent to refresh a specific action, when using the **-i** startup option or when a refresh interval is coded.

Tip: For an example of AT-TLS policy definitions see `/usr/lpp/tcpip/samples/pagent_TTLS.conf`

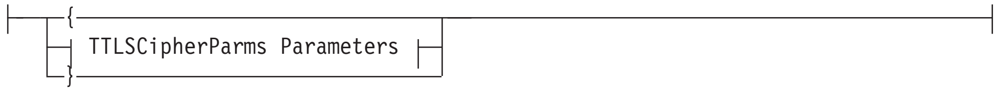
TTLSCipherParms statement

Use the TTLSCipherParms statement to define the cipher specifications for an AT-TLS environment or an AT-TLS connection. A TTLSCipherParms statement can be specified inline in a TTLSEnvironmentAction or TTLSConnectionAction statement or referenced by a TTLSEnvironmentAction or TTLSConnectionAction statement.

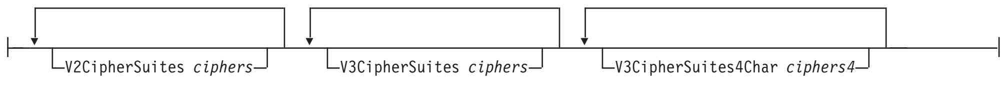
Syntax



Put Braces and Parameters on Separate Lines:



TTLSCipherParms Parameters:



Parameters

name

A string 1 - 32 characters in length specifying the name of this TTLSCipherParms statement.

Rule: If this TTLSCipherParms statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline TTLSCipherParms statement, a nonpersistent system name is created.

V2CipherSuites

Specifies the SSL version 2 cipher suites in order of preference. If a V2CipherSuites parameter is specified more than once, the values are concatenated to create a single list of cipher suites. For System SSL, the GSK_V2_CIPHER_SPECS value is set to the concatenated value.

The *ciphers* value is a string of one or more 1-character SSL version 2 ciphers or a single cipher constant. The cipher string cannot have blanks between each SSL version 2 cipher. If duplicate ciphers are specified, the first instance is used and all other instances are ignored. The maximum number of SSL version 2 ciphers is 10. For System SSL, see gsk_environment_open() in z/OS Cryptographic Services System SSL Programming for a list of valid cipher suites. Table 69 lists the supported cipher constants.

Table 69. V2CipherSuites

Cipher constant	Hexadecimal character
TLS_RC4_128_WITH_MD5	1
TLS_RC4_128_EXPORT40_WITH_MD5	2
TLS_RC2_CBC_128_CBC_WITH_MD5	3
TLS_RC2_CBC_128_CBC_EXPORT40_WITH_MD5	4
TLS_DES_64_CBC_WITH_MD5	6
TLS_DES_192_EDE3_CBC_WITH_MD5	7

V3CipherSuites

Specifies the SSL *ciphers* Version 3, TLS Version 1.0, TLS Version 1.1, or TLS Version 1.2 cipher suites in order of preference. If a V3CipherSuites or V3CipherSuites4Char parameter is specified more than once, the values are concatenated to create a single list of cipher suites. For System SSL, the GSK_V3_CIPHER_SPECS_EXPANDED value is set to the concatenated value.

The *ciphers* value is a string of one or more 2-hexadecimal character SSL *ciphers* Version 3, TLS version 1.0, TLS Version 1.1, or TLS Version 1.2 ciphers or a single cipher constant. The cipher string cannot have blanks between each SSL *ciphers* Version 3, TLS version 1.0, TLS Version 1.1, or TLS Version 1.2 cipher. If the string notation is used, you cannot specify any cipher values that require four character representation. Use the V3CipherSuites4Char parameter to specify four character cipher string values. If duplicate ciphers are specified, the first instance is used and all other instances ignored. The maximum number of ciphers that can be specified is 255. For System SSL, see gsk_environment_open() in z/OS Cryptographic Services System SSL Programming for a list of valid cipher suites. Table 70 lists the supported cipher constants.

V3CipherSuites4Char

Specifies the SSL *ciphers* Version 3, TLS Version 1.0, TLS Version 1.1, or TLS Version 1.2 cipher suites in order of preference. If a V3CipherSuites or V3CipherSuites4Char parameter is specified more than once, the values are concatenated to create a single list of cipher suites. For System SSL, the GSK_V3_CIPHER_SPECS_EXPANDED value is set to the concatenated value.

The *ciphers* value is a string of one or more 4-hexadecimal character SSL *ciphers* Version 3, TLS version 1.0, TLS Version 1.1, or TLS Version 1.2 ciphers. The cipher string cannot have blanks between each SSL *ciphers* Version 3, TLS version 1.0, TLS Version 1.1, or TLS Version 1.2 cipher. Use the V3CipherSuites parameter to specify a cipher constant or 2-character cipher string values. If duplicate ciphers are specified, the first instance is used and all other instances ignored. The maximum number of ciphers that can be specified is 255.

For System SSL, see gsk_environment_open() in z/OS Cryptographic Services System SSL Programming for a list of valid cipher suites. Table 70 lists the supported cipher constants.

Table 70. V3CipherSuites

Cipher constant	Hexadecimal character	Expanded character
TLS_NULL_WITH_NULL_NULL	00	0000
TLS_RSA_WITH_NULL_MD5	01	0001
TLS_RSA_WITH_NULL_SHA	02	0002
TLS_RSA_EXPORT_WITH_RC4_40_MD5	03	0003
TLS_RSA_WITH_RC4_128_MD5	04	0004
TLS_RSA_WITH_RC4_128_SHA	05	0005
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	06	0006
TLS_RSA_WITH_DES_CBC_SHA	09	0009
TLS_RSA_WITH_3DES_EDE_CBC_SHA	0A	000A
TLS_DH_DSS_WITH_DES_CBC_SHA	0C	000C
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	0D	000D
TLS_DH_RSA_WITH_DES_CBC_SHA	0F	000F

Table 70. V3CipherSuites (continued)

Cipher constant	Hexadecimal character	Expanded character
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	10	0010
TLS_DHE_DSS_WITH_DES_CBC_SHA	12	0012
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	13	0013
TLS_DHE_RSA_WITH_DES_CBC_SHA	15	0015
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	16	0016
TLS_RSA_WITH_AES_128_CBC_SHA	2F	002F
TLS_DH_DSS_WITH_AES_128_CBC_SHA	30	0030
TLS_DH_RSA_WITH_AES_128_CBC_SHA	31	0031
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	32	0032
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	33	0033
TLS_RSA_WITH_AES_256_CBC_SHA	35	0035
TLS_DH_DSS_WITH_AES_256_CBC_SHA	36	0036
TLS_DH_RSA_WITH_AES_256_CBC_SHA	37	0037
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	38	0038
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	39	0039
TLS_RSA_WITH_NULL_SHA256	3B	003B
TLS_RSA_WITH_AES_128_CBC_SHA256	3C	003C
TLS_RSA_WITH_AES_256_CBC_SHA256	3D	003D
TLS_DH_DSS_WITH_AES_128_CBC_SHA256	3E	003E
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	3F	003F
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	40	0040
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	67	0067
TLS_DH_DSS_WITH_AES_256_CBC_SHA256	68	0068
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	69	0069
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	6A	006A
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	6B	006B
TLS_RSA_WITH_AES_128_GCM_SHA256	9C	009C
TLS_RSA_WITH_AES_256_GCM_SHA384	9D	009D
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	9E	009E
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	9F	009F
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	A0	00A0
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	A1	00A1
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	A2	00A2
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	A3	00A3
TLS_DH_DSS_WITH_AES_128_GCM_SHA256	A4	00A4
TLS_DH_DSS_WITH_AES_256_GCM_SHA384	A5	00A5
TLS_ECDH_ECDSA_WITH_NULL_SHA		C001
TLS_ECDH_ECDSA_WITH_RC4_128_SHA		C002
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA		C003

Table 70. V3CipherSuites (continued)

Cipher constant	Hexadecimal character	Expanded character
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA		C004
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA		C005
TLS_ECDHE_ECDSA_WITH_NULL_SHA		C006
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA		C007
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA		C008
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA		C009
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA		C00A
TLS_ECDH_RSA_WITH_NULL_SHA		C00B
TLS_ECDH_RSA_WITH_RC4_128_SHA		C00C
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA		C00D
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA		C00E
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA		C00F
TLS_ECDHE_RSA_WITH_NULL_SHA		C010
TLS_ECDHE_RSA_WITH_RC4_128_SHA		C011
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA		C012
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA		C013
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA		C014
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256		C023
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384		C024
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256		C025
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384		C026
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256		C027
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384		C028
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256		C029
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384		C02A
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256		C02B
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384		C02C
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256		C02D
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384		C02E
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256		C02F
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384		C030
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256		C031
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384		C032

Requirement: If you plan to control access to the ICSF cryptographic support, TCP/IP and other applications must be permitted to access the ICSF/MVS cryptographic services (CSFSERV).

Guideline: If you do not have any reason to restrict access to the ICSF cryptographic support, you should not activate the CSFSERV resource class, define any of the profiles listed below, or permit any applications or users to these profiles. If you do need to set up controls in the CSFSERV resource class, enable the following resources.

Requirement: Elliptic Curve ciphers, defined as TLS_ECDH, TLS_ECDHE or TLS_ECDSA, require ICSF to be active.

The user ID running the AT-TLS application must have READ access to the following resources in the CSFSERV class:
- CSF1DVK
- CSF1GKP
- CSF1GAV
- CSF1PKS
- CSF1PKV
- CSF1TRC
- CSF1TRD

See Elliptic Curve Cryptography Support in z/OS Cryptographic Services System SSL Programming for additional information.

Requirement: AES-GCM ciphers require ICSF to be active.

The user ID running the AT-TLS application must have READ access to the following resources in the CSFSERV class:
- CSF1TRC
- CSF1SKD
- CSF1SKE
- CSF1TRD

TTLSConnectionAction statement

Use the TTLSConnectionAction statement to specify attributes for a subset of connections that need attributes different from those specified on the TTLSEnvironmentAction or TTLSGroupAction statement that is referenced by the same TTLSRule statement.

Syntax

```
►►──TTLSConnectionAction──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬──────────────────────────────────────┤
   ├─┤ TTLSConnectionAction Parameters ├─┤
   └─}──────────────────────────────┘
```

TTLSConnectionAction Parameters:

```
├──┬──────────────────────────────────────────┬──┬──────────────────────────┬──────►
   └─HandshakeRole──┬─Client─────────────┬──┘  ├─TTLSCipherParms─────────┤
                    ├─Server─────────────┤     └─TTLSCipherParmsRef  name─┘
                    └─ServerWithClientAuth─┘
```

```
►──┬──────────────────────────────┬──┬─────────────────────────┬──┬─────────┬──────►
   ├─TTLSSignatureParms───────────┤  └─CtraceClearText──┬─Off─┬─┘  └─Trace n─┘
   └─TTLSSignatureParmsRef  name──┘                     └─On──┘

►──┬────────────────────────────────────────┬──┬──────────────────────────┬──────────┤
   ├─TTLSConnectionAdvancedParms────────────┤  └─ConnectionUserInstance n─┘
   └─TTLSConnectionAdvancedParmsRef name────┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this TTLSConnectionAction statement.

HandshakeRole
Specifies the SSL handshake role to be taken. For System SSL, the GSK_SESSION_TYPE value is set to the same value as the HandshakeRole. If this value is specified on the TTLSConnectionAction statement, it is used instead of the value from the TTLSEnvironmentAction statement referenced by the same TTLSRule statement. Valid values are:

Client Perform the SSL handshake as a client.

Server Perform the SSL handshake as a server.

ServerWithClientAuth
Perform the SSL handshake as a server requiring client authentication.

TTLSCipherParms
An inline specification of a TTLSCipherParms statement. If this value is specified on the TTLSConnectionAction statement, it is used instead of the value from the TTLSEnvironmentAction statement referenced by the same TTLSRule statement.

TTLSCipherParmsRef
The name of a globally defined TTLSCipherParms statement. If this value is specified on the TTLSConnectionAction statement, it is used instead of the value from the TTLSEnvironmentAction statement referenced by the same TTLSRule statement.

TTLSSignatureParms
An inline specification of a TTLSSignatureParms statement. If this value is specified on the TTLSConnectionAction statement, it is used instead of the value from the TTLSEnvironmentAction statement that is referenced by the same TTLSRule statement.

TTLSSignatureParmsRef
The name of a globally defined TTLSSignatureParms statement. If this value is specified on the TTLSConnectionAction statement, it is used instead of the value from the TTLSEnvironmentAction statement that is referenced by the same TTLSRule statement.

CtraceClearText
Specifies whether application data traced using Ctrace or data trace are shown as unencrypted data. This parameter is applied only to connections that have active AT-TLS security on the connection. If this value is specified on the TTLSConnectionAction statement, it is used instead of the value from the TTLSEnvironmentAction or TTLSGroupAction statement referenced by the same TTLSRule statement. Valid values are:

Off Application data is not traced as clear text.

On Application data is traced as clear text.

`Trace`

Specifies the level of Application Transparent Transport Layer Security (AT-TLS) tracing. The valid values for *n* are in the range 0 - 255. The sum of the numbers associated with each level of tracing selected is the value that should be specified as *n*. If *n* is an odd number, errors are written to joblog, and all other configured traces are sent to syslogd. If this value is specified on the TTLSConnectionAction statement, it is used instead of the value from the TTLSEnvironmentAction or TTLSGroupAction statement referenced by the same TTLSRule statement.

0 No tracing is enabled.

1 (Error)
Errors are traced to the TCP/IP joblog.

2 (Error)
Errors are traced to syslogd. The messages are issued with syslogd priority code **err**.

4 (Info)
Tracing of when a connection is mapped to an AT-TLS rule and when a secure connection is successfully initiated is enabled. The messages are issued with syslogd priority code **info**.

8 (Event)
Tracing of major events is enabled. The messages are issued with syslogd priority code **debug**.

16 (Flow)
Tracing of system SSL calls is enabled. The messages are issued with syslogd priority code **debug**.

32 (Data)
Tracing of encrypted negotiation and headers is enabled. This traces the negotiation of secure sessions. The messages are issued with syslogd priority code **debug**.

64 Reserved.

128 Reserved.

255 All tracing is enabled.

`TTLSConnectionAdvancedParms`

An inline specification of a TTLSConnectionAdvancedParms statement.

`TTLSConnectionAdvancedParmsRef`

The name of a globally defined TTLSConnectionAdvancedParms statement.

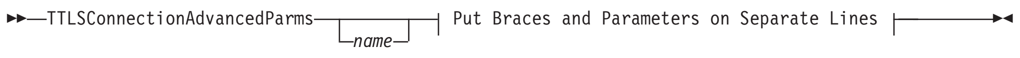
`ConnectionUserInstance`

Defines a configurable instance identifier for this TTLSConnectionAction statement. The *n* value can be in the range 0 - 65535. This parameter can be used to signal a change to the Policy Agent without modifying any of the other AT-TLS configuration statements. This parameter can also be used as a field to be updated when a change is made to this TTLSConnectionAction statement. This enables the user to differentiate TTLSConnectionAction statements, based on the instance identifier.

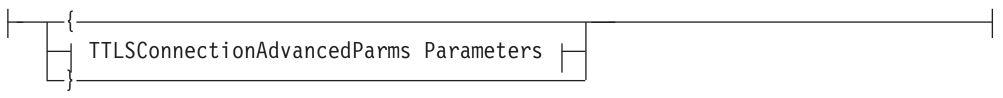
TTLSConnectionAdvancedParms statement

Use the TTLSConnectionAdvancedParms statement to specify attributes for a subset of connections that need attributes different from those specified on the TTLSEnvironmentAdvancedParms statement that is referenced by the same TTLSRule statement.

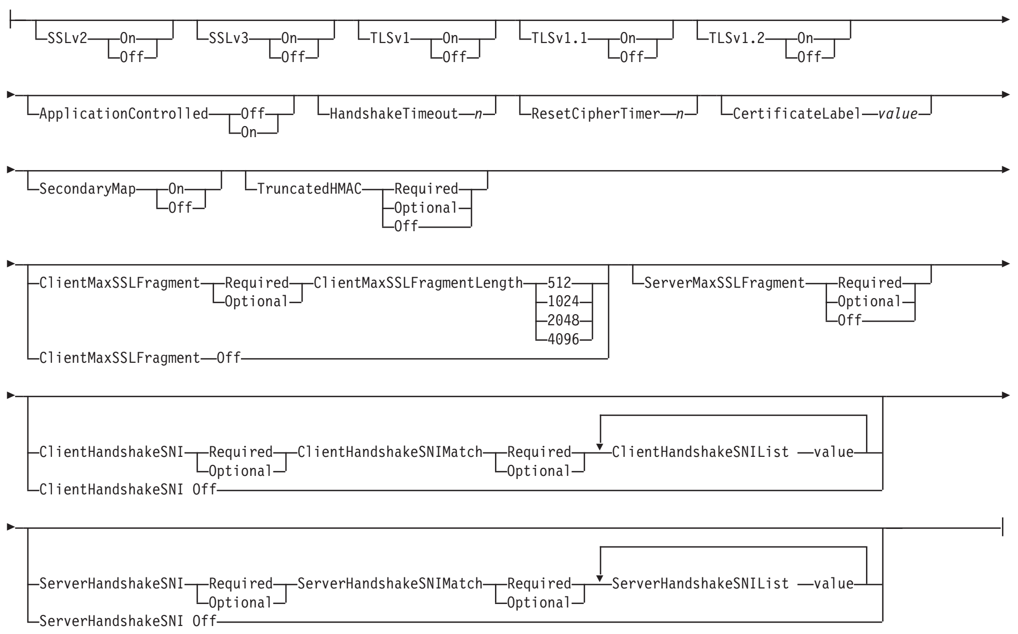
Syntax


TTLSConnectionAdvancedParms
name
Put Braces and Parameters on Separate Lines

Put Braces and Parameters on Separate Lines:



TTLSConnectionAdvancedParms Parmeters:


SSLv2 On Off
SSLv3 On Off
TLSv1 On Off
TLSv1.1 On Off
TLSv1.2 On Off
ApplicationControlled Off On
HandshakeTimeout n
ResetCipherTimer n
CertificateLabel value
SecondaryMap On Off
TruncatedHMAC Required Optional Off
ClientMaxSSLFragment Required Optional ClientMaxSSLFragmentLength 512 1024 2048 4096
ClientMaxSSLFragment Off
ServerMaxSSLFragment Required Optional Off
ClientHandshakeSNI Required Optional ClientHandshakeSNIMatch Required Optional ClientHandshakeSNIList value
ClientHandshakeSNI Off
ServerHandshakeSNI Required Optional ServerHandshakeSNIMatch Required Optional ServerHandshakeSNIList value
ServerHandshakeSNI Off

Parameters

name
: A string 1 - 32 characters in length specifying the name of this TTLSConnectionAdvancedParms statement.

  Rule: If this TTLSConnectionAdvancedParms statement is not specified inline within another statement, a *name* value must be provided. If a name value is not specified for an inline TTLSConnectionAdvancedParms statement, a nonpersistent system name is created.

SSLv2
: Specifies the state of the SSL Version 2 protocol. For System SSL, the GSK_PROTOCOL_SSLV2 value is set to this value. Possible values are:

On Enables the SSL Version 2 protocol.

Off Disable the SSL Version 2 protocol.

SSLv3
Specifies the state of the SSL Version 3 protocol. For System SSL, the GSK_PROTOCOL_SSLV3 value is set to this value. Possible values are:

On Enable the SSL Version 3 protocol.

Off Disable the SSL Version 3 protocol.

TLSv1
Specifies the state of the TLS Version 1 protocol. For System SSL, the GSK_PROTOCOL_TLSV1 value is set to this value. Possible values are:

On Enable the TLS Version 1.0 protocol.

Off Disable the TLS Version 1.0 protocol.

TLSv1.1
Specifies the state of the TLS version 1.1 protocol. For System SSL, the GSK_PROTOCOL_TLSV1.1 value is set to this value. Possible values are:

On Enable the TLS Version 1.1 protocol.

Off Disable the TLS Version 1.1 protocol.

TLSv1.2
Specifies the state of the TLS version 1.2 protocol. For System SSL, the GSK_PROTOCOL_TLSV1.2 value is set to this value. Possible values are:

On Enable the TLS Version 1.2 protocol.

Off Disable the TLS Version 1.2 protocol.

TruncatedHMAC
For TLSv1.0 protocol or later, this keyword specifies whether clients and servers support the use of 80-bit truncated HMACs. For System SSL, the extension ID is set to GSK_TLS_SET_TRUNCATED_HMAC and a flag is set in the gsk_tls_extension structure if it is required. Possible values are:

Required
Specifies that 80-bit truncated HMAC support must be accepted by both endpoints. Connections fail if the remote endpoint does not support the 80-bit truncated HMAC.

Tip: When you specify TruncatedHMAC as Required, specify SSLv3 as Off.

Optional
Specifies that support is provided for 80-bit truncated HMAC negotiation, but connections with endpoints that do not support the truncated 80-bit HMAC are allowed.

Off Specifies that support is not provided for 80-bit truncated HMAC negotiation. The function is not enabled. Connections fail if the remote endpoint requires support for the 80-bit truncated HMAC.

ClientMaxSSLFragment
For TLSv1.0 protocol or later, this keyword specifies whether the maximum SSL fragment function is supported when AT-TLS is the TLS client on the connection. For System SSL, the extension ID is set to GSK_TLS_SET_CLIENT_MFL and a flag is set in the gsk_tls_extension structure if it is required. Possible values are:

Required
Specifies that maximum SSL fragment function support must be accepted by the server. Connections fail if the server does not support the maximum SSL fragment function.

Tip: When you specify ClientMaxSSLFragment as Required, specify SSLv3 as Off.

Optional
Specifies support for maximum SSL fragment function negotiation, but allows connections with servers that do not support maximum SSL fragment function.

Off Specifies that maximum SSL fragment function negotiation is not supported. The function is not enabled. Connections fail if the server requires support for maximum SSL fragment function.

`ClientMaxSSLFragmentLength`
Specifies the maximum SSL fragment function, in bytes, to request on the connection when AT-TLS is the TLS client using TLSv1.0 protocol or later. The valid values are 512, 1024, 2048, and 4096. For System SSL, the maximum fragment function is set to GSK_TLS_MFL_512, GSK_TLS_MFL_1024, GSK_TLS_MFL_2048, or GSK_TLS_MFL_4096. This parameter is required when ClientMaxSSLFragment is set to Required or Optional.

`ServerMaxSSLFragment`
For TLSv1.0 protocol or later, this keyword specifies whether the maximum SSL fragment function is supported when AT-TLS is the TLS server on the connection. For System SSL, the extension ID is set to GSK_TLS_SET_SERVER_MFL and a flag is set in the gsk_tls_extension structure if it is required. Possible values are:

Required
Specifies that maximum SSL fragment function support must be accepted by the client. Connections fail if the client does not support the maximum SSL fragment function.

Tip: When you specify ServerMaxSSLFragment as Required, specify SSLv3 as Off.

Optional
Specifies that support is provided for maximum SSL fragment function, but allow connections with clients that do not support the maximum SSL fragment function.

Off Specifies that maximum SSL fragment function is not supported. The function is not enabled. Connections fail if the client requires support for maximum SSL fragment function.

`ClientHandshakeSNI`
For TLSv1.0 protocol or later, this keyword specifies whether a client can specify a list of server names. The server chooses a certificate based on that server name list for this connection. For System SSL, the extension ID is set to GSK_TLS_SET_SNI_CLIENT_SNAMES and a flag is set in the gsk_tls_extension structure if it is required. Valid values are:

Required
Specifies that server name indication support must be accepted by the server. Connections fail if the server does not support server name indication.

Tip: When you specify ClientHandshakeSNI as required, specify SSLv3 as Off.

Optional
Specifies that server name indication negotiation is supported, but allows connections with servers that do not support server name indication negotiation.

Off Specifies that server name indication is not supported. The function is not enabled. Connections fail if the server requires support for server name indication. This is the default.

`ClientHandshakeSNIMatch`
Code this parameter if ClientHandshakeSNI is set to Required or Optional. For system SSL, a flag is set in the gsk_sni_client_snames structure if a match is required. Possible values are:

Required
Specifies that a server name in the list of server names provided by the TLS client must match a server name in the list of server names and certificate labels on the TLS server. The connection ends if no match was found for the server name at the server.

Optional
Specifies that connections can continue if no match is found for the server name.

`ClientHandshakeSNIList`
For SSL clients using TLSv1.0 protocol or later, this keyword specifies a server name. You can code multiple ClientHandshakeSNIList statements. The list of server names is passed to the server in the SSL handshake. For System SSL, the server names are anchored in the gsk_sni_client_snames structure. A server name can be 1 - 255 characters in length. This parameter is required when ClientHandshakeSNI is set to Required or Optional.

Restriction: The total length of all the server names specified must be less than 32K.

`ServerHandshakeSNI`
For TLSv1.0 protocol or later, this keyword specifies whether a certificate is chosen based on the server name list provided by the TLS client. For System SSL, the extension ID is set to GSK_TLS_SET_SNI_SERVER_SNAMES and a flag is set in the gsk_tls_extension structure if it is required. Possible values are:

Required
Specifies that server name indication support must be accepted by the client. Connections fail if the client does not support server name indication.

Tip: When you specify ServerHandshakeSNI as Required, specify SSLv3 as Off.

Optional
Specifies that server name indication negotiation is supported, but allow connections with clients that do not support server name indication.

Off Specifies that server name indication is not supported. The function is not enabled. Connections fail if the client requires support for server name indication.

ServerHandshakeSNIMatch
You must code this parameter when ServerHandshakeSNI is set to Required or Optional. For system SSL, a flag is set in the gsk_sni_server_labels structure if a match is required. Possible values are:

Required
Specifies that a server name in the list of server names provided by the TLS client must match a server name in the ServerHandshakeSNIList. The connection ends if no match can be found for the server name.

Optional
Specifies that connections to continue if no match is found for the server name.

ServerHandshakeSNIList
For SSL servers using TLSv1.0 protocol or later, this keyword specifies a server name and certificate label pair to be used by the server, separated by a slash (/). Multiple ServerHandshakeSNIList statements can be coded. The server matches the server name provided by the client to a certificate label. For System SSL, the server names and labels are anchored in the gsk_sni_server_labels structure. A server name can be 1 - 255 characters in length. A certificate label can be 1 - 127 characters in length. This parameter is required when ServerHandshakeSNI is set to Required or Optional.

Rule: You can use comment indicators and embedded blanks as part of the certificate label value for this attribute. For example:

```
ServerHandshakeSNIList myservername/Root#CA  Certificate
value used: myservername/Root#CA  Certificate
```

Restrictions:

- The total length of all the server names and certificate labels specified must be less than 32K.
- When the certificate label value contains embedded blanks, you must specify the entire parameter value within the first 1 536 characters of the configuration file line.

ApplicationControlled
Specifies whether the application can control AT-TLS security for a connection. Valid values are:

Off An application cannot control AT-TLS security. The connection automatically negotiates AT-TLS security.

On An application can control AT-TLS security. AT-TLS security is used only when requested by the application, using the SIOCTTLSCTL ioctl.

HandshakeTimeout
Specifies the number of seconds to wait for the initial handshake to complete. Valid values of *n* are in the range 0 - 600.

For connections with the HandshakeRole parameter set to Client, the timer is initially set to 5 times the value of *n*, allowing for network delay and any delay on the server in processing the connection. When the initial response is received from the server, the timer is set again for *n* seconds, to allow the initial handshake to complete.

For connections with the HandshakeRole parameter set to Server or ServerWithClientAuth, when the server starts to process the new connection the timer is set to *n* seconds, waiting for the initial request from the client. The timer is reset to *n* seconds when the server sends the initial response, to allow the initial handshake to complete.

If the timer expires, the TCP connection is reset. A value of 0 indicates that the connection does not time out waiting for the initial handshake to complete.

`ResetCipherTimer`
Specifies the number of minutes a secure connection can be active before a new session key is generated for the connection. AT-TLS initiates a handshake on the next read or write after the timer expires. For System SSL, the GSK_RESET_CIPHER function is used to initiate handshake. This timer applies only to connections using SSLv3 or TLSv1 protocol. If the session ID has expired, a full handshake is performed. Otherwise, a short handshake is performed. For System SSL, session expiration is controlled by the GSK_V3_SESSION_TIMEOUT statement. Valid values of *n* are in the range 0 - 1440. Specifying 0 means that the session key refresh is not initiated by AT-TLS for the life of the connection.

`CertificateLabel`
Specifies the label of the certificate to be used for authentication. Valid values are in the range 1 - 127 characters in length. For System SSL, the GSK_KEYRING_LABEL value is set to this value.

Rule: Comment indicators and embedded blanks are treated as part of the value for this attribute. For example:

```
CertificateLabel  Root#CA  Certificate
value used:   Root#CA  Certificate
```

Restriction: When the value contains embedded blanks, you must specify the entire value within the first 1 536 characters of the configuration file line.

`SecondaryMap`
Specifies whether the application establishes secondary connections that should use the secondary policy mapping method. When specified in the TTLSConnectionAdvancedParms, this statement overrides the values specified in the TTLSEnvironmentAdvancedParms and TTLSGroupAdvancedParms. Valid values are:

Off A connection that maps to this policy should not be used as a primary connection in the secondary policy mapping method.

On A connection that maps to this policy should be used as a primary connection in the secondary policy mapping method. Future connections established between the same two IP addresses by the same process that do not map to any policy or map to a policy with a lower priority are considered secondary connections. These secondary connections use the same policy mapped by the associated primary connection.

TTLSEnvironmentAction statement

Use the TTLSEnvironmentAction statement to specify the attributes for an AT-TLS environment. A TTLSEnvironmentAction statement is required if the TTLSGroupAction statement, referenced on the same TTLSRule statement, specifies TTLSEnabled as **On**.

Syntax

```
►►──TTLSEnvironmentAction──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬──────────────────────────────────────┤
   ├─┤ TTLSEnvironmentAction Parameters ├─┤
   └─}──────────────────────────────────┘
```

TTLSEnvironmentAction Parameters:

```
├──┬─TTLSKeyringParms──────────┬──HandshakeRole──┬─Client──────────────┬──────────►
   └─TTLSKeyringParmsRef  name─┘                 ├─Server──────────────┤
                                                 └─ServerWithClientAuth─┘

   ┌─SuiteBProfile──Off────┐
►──┼───────────────────────┼──┬──────────────────────────┬──────────────────────►
   └─SuiteBProfile──┬─Off──┘  └─┬─TTLSCipherParms────────┬┘
                    ├─128─┤     └─TTLSCipherParmsRef  name┘
                    ├─192─┤
                    └─All─┘

►──┬─────────────────────────────┬──┬──────────────────────────┬──┬─────────┬───►
   ├─TTLSSignatureParms──────────┤  └─CtraceClearText──┬─Off─┬─┘  └─Trace n─┘
   └─TTLSSignatureParmsRefname───┘                     └─On──┘

►──┬──────────────────────────────────────┬──┬──────────────────────────────┬───►
   ├─TTLSEnvironmentAdvancedParms─────────┤  ├─TTLSGskAdvancedParms─────────┤
   └─TTLSEnvironmentAdvancedParmsRefname──┘  └─TTLSGskAdvancedParmsRef name─┘

►──┬──────────────────────────┬─────────────────────────────────────────────────┤
   └─EnvironmentUserInstance n┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this TTLSEnvironmentAction statement.

TTLSKeyringParms
: An inline specification of a TTLSKeyringParms statement. This is a required parameter.

TTLSKeyringParmsRef
: The name of a globally defined TTLSKeyringParms statement.

HandshakeRole
: Specifies the SSL handshake role to be taken for connections in this AT-TLS environment. For System SSL, the GSK_SESSION_TYPE value is set to the same value as the HandshakeRole. This is a required parameter. Valid values are:

  Client Perform the SSL handshake as a client.

  Server Perform the SSL handshake as a server.

  ServerWithClientAuth
  Perform the SSL handshake as a server requiring client authentication.

`SuiteBProfile`
Specified the RFC5430 Suite B cipher suites to apply to TLS sessions. For System SSL, the GSK_SUITE_B_PROFILE value is set to the value of SuiteBProfile. Valid values are:

Off The use of TLS V1.2 and Suite B cipher suites is not required. This is the default.

128 Suite B 128 bit cipher suites will be used.

192 Suite B 192 bit cipher suites will be used.

All All Suite B cipher suites will be used.

Result: When 128, 192, or All is coded, any TTLSCipherParms statements are ignored. Only the ciphers that are defined in the Suite B profile will be used.

`TTLSCipherParms`
An inline specification of a TTLSCipherParms statement.

`TTLSCipherParmsRef`
The name of a globally defined TTLSCipherParms statement.

`TTLSSignatureParms`
An inline specification of a TTLSSignatureParms statement.

`TTLSSignatureParmsRef`
The name of a globally defined TTLSSignatureParms statement.

`CtraceClearText`
Specifies whether application data traced using Ctrace or data trace is shown as unencrypted data. This parameter is applied only to connections that have active AT-TLS security on the connection. If this value is specified on the TTLSEnvironmentAction statement, it is used instead of the value from the TTLSGroupAction statement referenced by the same TTLSRule statement. Valid values are:

Off Application data is not traced as clear text.

On Application data is traced as clear text.

`Trace`
Specifies the level of AT-TLS tracing. The valid values for *n* are in the range 0 - 255. The sum of the numbers associated with each level of tracing selected is the value that should be specified as *n*. If *n* is an odd number, errors are written to joblog and all other configured traces are sent to syslogd. If this value is specified on the TTLSEnvironmentAction statement, it is used instead of the value from the TTLSGroupAction statement referenced by the same TTLSRule statement.

0 No tracing is enabled.

1 (Error)
Errors are traced to the TCP/IP joblog

2 (Error)
Errors are traced to syslogd. The messages are issued with syslogd priority code **err**.

4 (Info)
Tracing of when a connection is mapped to an AT-TLS rule and when a secure connection is successfully initiated is enabled. The messages are issued with syslogd priority code **info**.

8 (Event)
: Tracing of major events is enabled. The messages are issued with syslogd priority code **debug**.

16 (Flow)
: Tracing of system SSL calls is enabled. The messages are issued with syslogd priority code **debug**.

32 (Data)
: Tracing of encrypted negotiation and headers is enabled. This traces the negotiation of secure sessions. The messages are issued with syslogd priority code **debug**.

64 Reserved.

128 Reserved.

255 All tracing is enabled.

`TTLSEnvironmentAdvancedParms`
An inline specification of a TTLSEnvironmentAdvancedParms statement.

`TTLSEnvironmentAdvancedParmsRef`
The name of a globally defined TTLSEnvironmentAdvancedParms statement.

`TTLSGskAdvancedParms`
An inline specification of a TTLSGskAdvancedParms statement.

`TTLSGskAdvancedParmsRef`
The name of a globally defined TTLSGskAdvancedParms statement.

`EnvironmentUserInstance`
Defines a configurable instance identifier for this TTLSEnvironmentAction statement. The *n* value can be in the range 0 - 65535. This parameter can be used to signal a change to the Policy Agent without modifying any of the other AT-TLS configuration statements. For example, when the contents of the key ring has changed, but the key ring name is unchanged. Adding or updating the EnvironmentUserInstance parameter would signal Policy Agent to install a new TTLSEnvironmentAction statement. This parameter can also be used as a field to be updated when a change is made to this TTLSEnvironmentAction statement. This enables the user to differentiate TTLSEnvironmentAction statements, based on the instance identifier.

TTLSEnvironmentAdvancedParms statement

Use the TTLSEnvironmentAdvancedParms statement to specify advanced attributes for an AT-TLS environment.

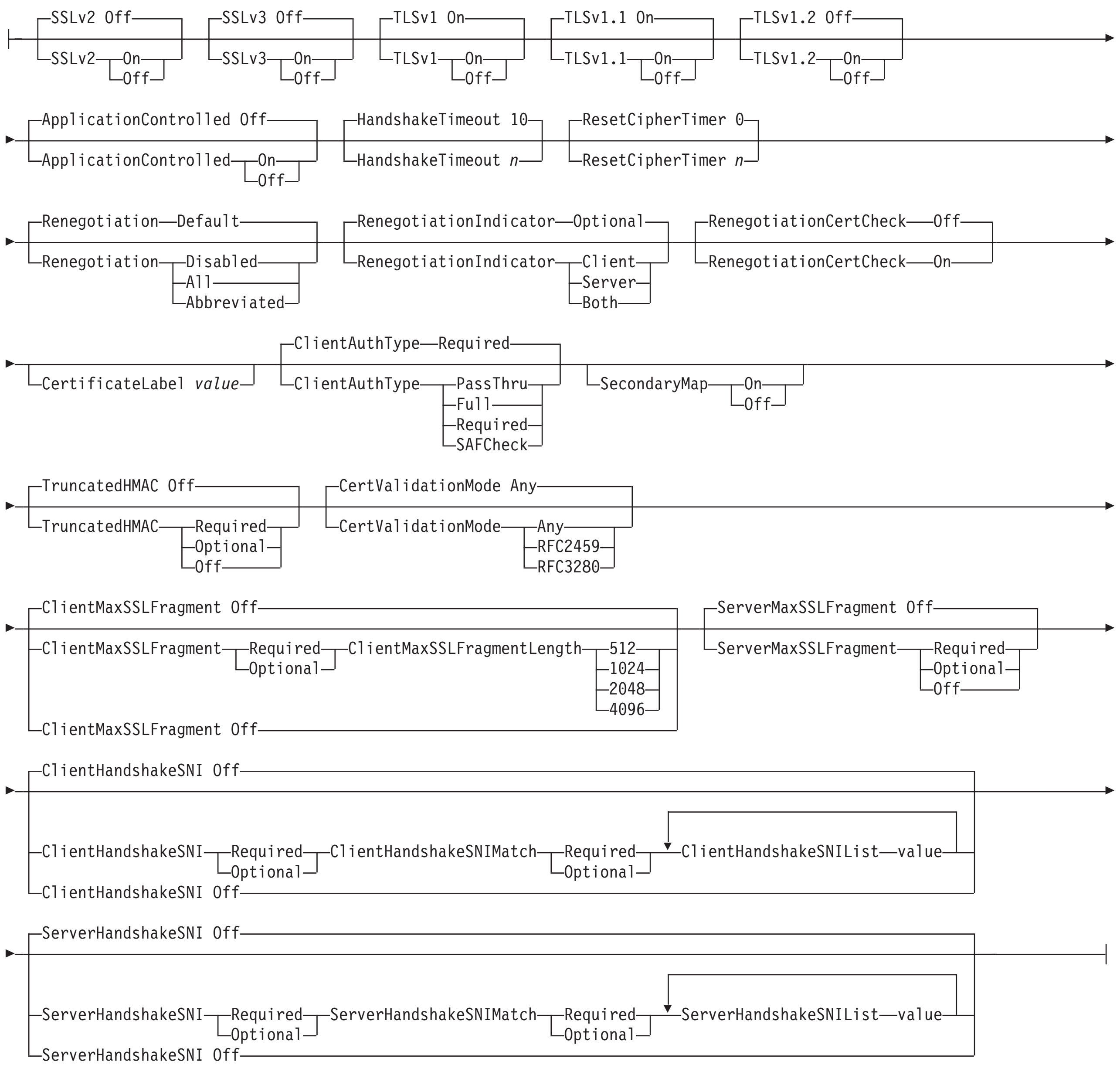
Syntax

```
►►──TTLSEnvironmentAdvancedParms──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                                  └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────────┬──────────────────────────────────────────────┤
   ├──┤ TTLSEnvironmentAdvancedParms Parameters ├──┤
   └─}──────────────────────────────────────┘
```

TTLSEnvironmentAdvancedParms Parameters:



SSLv2 Off
SSLv2 On Off
SSLv3 Off
SSLv3 On Off
TLSv1 On
TLSv1 On Off
TLSv1.1 On
TLSv1.1 On Off
TLSv1.2 Off
TLSv1.2 On Off
ApplicationControlled Off
ApplicationControlled On Off
HandshakeTimeout 10
HandshakeTimeout n
ResetCipherTimer 0
ResetCipherTimer n
Renegotiation Default
Renegotiation Disabled All Abbreviated
RenegotiationIndicator Optional
RenegotiationIndicator Client Server Both
RenegotiationCertCheck Off
RenegotiationCertCheck On
CertificateLabel value
ClientAuthType Required
ClientAuthType PassThru Full Required SAFCheck
SecondaryMap On Off
TruncatedHMAC Off
TruncatedHMAC Required Optional Off
CertValidationMode Any
CertValidationMode Any RFC2459 RFC3280
ClientMaxSSLFragment Off
ClientMaxSSLFragment Required Optional ClientMaxSSLFragmentLength 512 1024 2048 4096
ClientMaxSSLFragment Off
ServerMaxSSLFragment Off
ServerMaxSSLFragment Required Optional Off
ClientHandshakeSNI Off
ClientHandshakeSNI Required Optional ClientHandshakeSNIMatch Required Optional ClientHandshakeSNIList value
ClientHandshakeSNI Off
ServerHandshakeSNI Off
ServerHandshakeSNI Required Optional ServerHandshakeSNIMatch Required Optional ServerHandshakeSNIList value
ServerHandshakeSNI Off

Parameters

name

A string 1 - 32 characters in length specifying the name of this TTLSEnvironmentAdvancedParms statement.

Rule: If this TTLSEnvironmentAdvancedParms statement is not specified inline within another statement, a *name* value must be provided. If a name value is not specified for an inlineTTLSEnvironmentAdvancedParms statement, a nonpersistent system name is created.

SSLv2

Specifies the state of the SSL Version 2 protocol. For System SSL, the GSK_PROTOCOL_SSLV2 value is set to this value. Possible values are:

On Enables the SSL Version 2 protocol.

Off Disables the SSL Version 2 protocol. This is the default.

SSLv3
Specifies the state of the SSL Version 3 protocol. For System SSL, the GSK_PROTOCOL_SSLV3 value is set to this value. Possible values are:

| **On** Enable the SSL Version 3 protocol.

| **Off** Disable the SSL Version 3 protocol. This is the default.

TLSv1
Specifies the state of the TLS Version 1 protocol. For System SSL, the GSK_PROTOCOL_TLSV1 value is set to this value. Possible values are:

On Enable the TLS Version 1.0 protocol. This is the default.

Off Disable the TLS Version 1.0 protocol.

TLSv1.1
Specifies the state of the TLS Version 1.1 protocol. For System SSL, the GSK_PROTOCOL_TLSV1_1 value is set to this value. Possible values are:

On Enable the TLS Version 1.1 protocol. This is the default.

Off Disable the TLS Version 1.1 protocol.

TLSv1.2
Specifies the state of the TLS Version 1.2 protocol. For System SSL, the GSK_PROTOCOL_TLSV1_2 value is set to this value. Possible values are:

On Enable the TLS Version 1.2 protocol.

Tip: When you specify TLSv1.2 as On, System SSL will not negotiate SSLv2 sessions even if you specify SSLv2 as On.

Off Disable the TLS Version 1.2 protocol. This is the default.

CertValidationMode
Specifies the method of certificate validation. For System SSL, the GSK_CERT_VALIDATION_MODE value is set to this value. Possible values are:

Any Specifies that certificate validation can use any supported X.509 certificate validation method. This is the default.

RFC2459
Specifies that certificates are validated using the method described in RFC 2459.

RFC3280
Specifies that certificates are validated using the method described in RFC 3280.

TruncatedHMAC
For TLSv1.0 protocol and later, this keyword specifies whether clients and servers support the use of 80-bit truncated HMACs. For System SSL, the extension ID is set to GSK_TLS_SET_TRUNCATED_HMAC and a flag is set in the gsk_tls_extension structure, if it is required. Possible values are:

Required
Specifies that 80-bit truncated HMAC support must be accepted by both endpoints. Connections fail if the remote endpoint does not support the 80-bit truncated HMAC.

Tip: When you specify TruncatedHMAC as Required, specify SSLv3 as Off.

Optional
: Specifies that support is provided for 80-bit truncated HMAC negotiation, but connections with endpoints that do not support the truncated 80-bit HMAC are allowed.

Off
: Specifies that support is not provided for 80-bit truncated HMAC negotiation. The function is not enabled. Connections fail if the remote endpoint requires support for the 80-bit truncated HMAC. This is the default.

`ClientMaxSSLFragment`

For TLSv1.0 protocol and later, this keyword specifies whether maximum SSL fragment function is supported when AT-TLS is the TLS client on the connection. For System SSL, the extension ID is set to GSK_TLS_SET_CLIENT_MFL and a flag is set in the gsk_tls_extension structure if it is required. Possible values are:

Required
: Specifies that maximum SSL fragment function support must be accepted by the server. Connections fail if the server does not support maximum SSL fragment function.

Tip: When you specify ClientMaxSSLFragment as Required, specify SSLv3 as Off.

Optional
: Specifies support for maximum SSL fragment function negotiation, but allows connections with servers that do not support maximum SSL fragment function.

Off
: Specifies that maximum SSL fragment function negotiation is not supported. The function is not enabled. Connections fail if the server requires support for maximum SSL fragment function. This is the default.

`ClientMaxSSLFragmentLength`

For TLSv1.0 protocol and later, this value specifies maximum SSL fragment function, in bytes, to request on the connection when AT-TLS is the TLS client using TLSv1.0 and TLSv1.1 protocols. The valid values are 512, 1024, 2048, and 4096. For System SSL, the maximum fragment length is set to GSK_TLS_MFL_512, GSK_TLS_MFL_1024, GSK_TLS_MFL_2048, or GSK_TLS_MFL_4096. This parameter is required when ClientMaxSSLFragment is set to Required or Optional.

`ServerMaxSSLFragment`

For TLSv1.0 protocol and later, this keyword specifies whether the maximum SSL fragment function is supported when AT-TLS is the TLS server on the connection. For System SSL, the extension ID is set to GSK_TLS_SET_SERVER_MFL and a flag is set in the gsk_tls_extension structure if it is required. Possible values are:

Required
: Specifies that maximum SSL fragment function support must be accepted by the client. Connections fail if the client does not support maximum SSL fragment function.

Tip: When you specify ServerMaxSSLFragment as Required, specify SSLv3 as Off.

Optional
Specifies that support is provided for maximum SSL fragment function, but allow connections with clients that do not support maximum SSL fragment function.

Off Specifies that maximum SSL fragment function is not supported. The function is not enabled. Connections fail if the client requires support for maximum SSL fragment function. This is the default value.

ClientHandshakeSNI
For TLSv1.0 protocol and later, this keyword specifies whether a client can specify a list of server names. The server chooses a certificate based on that server name list for this connection. For System SSL, the extension ID is set to GSK_TLS_SET_SNI_CLIENT_SNAMES and a flag is set in the gsk_tls_extension structure if it is required. Valid values are:

Required
Specifies that server name indication support must be accepted by the server. Connections fail if the server does not support server name indication.

Tip: When you specify ClientHandshakeSNI as required, specify SSLv3 as Off.

Optional
Specifies that server name indication negotiation is supported, but allows connections with servers that do not support server name indication negotiation.

Off Specifies that server name indication is not supported. The function is not enabled. Connections fail if the server requires support for server name indication. This is the default.

ClientHandshakeSNIMatch
Code this parameter if ClientHandshakeSNI is set to Required or Optional. For system SSL, a flag is set in the gsk_sni_client_snames structure if a match is required. Possible values are:

Required
Specifies that a server name in the list of server names provided by the TLS client must match a server name in the list of server names and certificate labels on the TLS server. The connection ends if no match was found for the server name at the server.

Optional
Specifies that connections can continue if no match is found for the server name.

ClientHandshakeSNIList
For SSL clients using TLSv1.0 protocol and later, this keyword specifies a server name. You can code multiple ClientHandshakeSNIList statements. The list of server names is passed to the server in the SSL handshake. For System SSL, the server names are anchored in the gsk_sni_client_snames structure. A server name can be 1 - 255 characters in length. This parameter is required when ClientHandshakeSNI is set to Required or Optional.

Restriction: The total length of all the server names specified must be less than 32K.

ServerHandshakeSNI
For TLSv1.0 protocol and later, this keyword specifies whether a certificate is chosen based on the server name list provided by the TLS client. For System

SSL, the extension ID is set to GSK_TLS_SET_SNI_SERVER_SNAMES and a flag is set in the gsk_tls_extension structure if it is required. Possible values are:

Required
Specifies that server name indication support must be accepted by the client. Connections fail if the client does not support server name indication.

Tip: When you specify ServerHandshakeSNI as Required, specify SSLv3 as Off.

Optional
Specifies that server name indication negotiation is supported, but allow connections with clients that do not support server name indication.

Off Specifies that server name indication is not supported. The function is not enabled. Connections fail if the client requires support for server name indication. This is the default value.

`ServerHandshakeSNIMatch`
You must code this parameter when ServerHandshakeSNI is set to Required or Optional. For system SSL, a flag is set in the gsk_sni_server_labels structure if a match is required. Possible values are:

Required
Specifies that a server name in the list of server names provided by the TLS client must match a server name in the ServerHandshakeSNIList . The connection ends if no match can be found for the server name.

Optional
Specifies that connections continue if no match is found for the server name.

`ServerHandshakeSNIList`
For SSL servers using TLSv1.0 protocol and later, this keyword specifies a server name and certificate label pair to be used by the server, separated by a slash (/). Multiple ServerHandshakeSNIList statements can be coded. The server matches the server name provided by the client to a certificate label. For System SSL, the server names and labels are anchored in the gsk_sni_server_labels structure. A server name can be 1 - 255 characters in length. A certificate label can be 1 - 127 characters in length. This parameter is required when ServerHandshakeSNI is set to Required or Optional.

Rule: You can use comment indicators and embedded blanks as part of the certificate label value for this attribute. For example:

```
ServerHandshakeSNIList myservername/Root#CA Certificate
value used: myservername/Root#CA Certificate
```

Restrictions:

- The total length of all the server names and certificate labels specified must be less than 32K.
- When the certificate label value contains embedded blanks, you must specify the entire parameter value within the first 1 536 characters of the configuration file line.

`ApplicationControlled`
Specifies whether the application can control AT-TLS security for a connection. Valid values are:

Off An application cannot control AT-TLS security. The connection automatically negotiates AT-TLS security. This is the default.

On An application can control AT-TLS security. AT-TLS security is used only when requested by the application, using the SIOCTTLSCTL ioctl.

HandshakeTimeout
Specifies the number of seconds to wait for the initial handshake to complete. Valid values of *n* are in the range 0 - 600. The default value is 10.

For connections with the HandshakeRole parameter set to Client, the timer is initially set to 5 times the value of *n*, allowing for network delay and any delay on the server in processing the connection. When the initial response is received from the server, the timer is set again for *n* seconds, to allow the initial handshake to complete.

For connections with that HandshakeRole parameter set to Server or ServerWithClientAuth, when the server starts to process the new connection the timer is set to *n* seconds, waiting for the initial request from the client. The timer is reset to *n* seconds when the server sends the initial response, to allow the initial handshake to complete.

If the timer expires, the TCP connection is reset. A value of 0 indicates that the connection does not time out waiting for the initial handshake to complete.

ResetCipherTimer
Specifies the number of minutes a secure connection can be active before a new session key is generated for the connection. AT-TLS initiates a handshake on the next read or write after the timer expires. For System SSL, the GSK_RESET_CIPHER function is used to initiate this. If the session ID has expired, controlled by the GSK_V3_SESSION_TIMEOUT statement, a full handshake is performed. Otherwise, a short handshake is performed. This timer applies only to connections using SSLv3 or TLSv1 protocol. Valid values of *n* are in the range 0 - 1440. Specifying 0 means that session key refresh is not initiated by AT-TLS for the life of the connection. The default value is 0.

Renegotiation
Specifies the type of session key renegotiation that is allowed. For System SSL, the GSK_RENEGOTIATION value is set. The following values are valid:

Default
GSK_RENEGOTIATION set to NONE. Disables SSL V3 and TLS handshake renegotiation as a server and allows RFC 5746 renegotiation. This is the default.

| **Disabled**
Disables SSL V3 and TLS handshake renegotiation as a server and disables RFC 5746 renegotiation.

All Allows SSL V3 and TLS handshake renegotiation as a server and allows RFC 5746 renegotiation.

Abbreviated
Allows SSL V3 and TLS abbreviated handshake renegotiation as a server for resuming the current session only, while disabling SSL V3 and TLS full handshake renegotiation as a server. The System SSL session ID cache is not checked when resuming the current session. Allows RFC 5746 renegotiation.

RenegotiationIndicator
Sets the enforcement level of the initial handshake renegotiation indication as

RFC 5746 specifies. For System SSL, the GSK_EXTENDED_RENEGOTIATION_INDICATOR value is set to this value. The following values are valid:

Optional
The renegotiation indicator is not required during initial handshake.

Client Allow the client initial handshake to proceed only when the server indicates support for RFC 5746 renegotiation.

Server Allow the server initial handshake to proceed only when the client indicates support for RFC 5746 renegotiation.

Both Allow the client and server initial handshakes to proceed only when the partner indicates support for RFC 5746 renegotiation.

`RenegotiationCertCheck`
Specifies whether to perform an identity check against the peer's certificate during renegotiation. For System SSL, the GSK_RENEGOTIATION_PEER_CERT_CHECK value is set to this value. Valid values are:

Off An identity check is not performed. This allows the peer certificate to change during renegotiation.

On An identity check is performed. This ensures that the peer certificate does not change during renegotiation.

`CertificateLabel`
Specifies the label of the certificate to be used for authentication. Valid values are in the range 1 - 127 characters in length. For System SSL, the GSK_KEYRING_LABEL value is set to this value.

Rule: Comment indicators and embedded blanks are treated as part of the value for this attribute. For example:

```
CertificateLabel  Root#CA  Certificate
value used:   Root#CA  Certificate
```

Restriction: When the value contains embedded blanks, you must specify the entire value within the first 1 536 characters of the configuration file line.

`ClientAuthType`
Specifies the type of client certificate validation to be performed for connections in this AT-TLS environment. Client certificates are requested only if HandshakeRole is set to **ServerWithClientAuth**. Valid values are:

PassThru
Bypasses client certificate validation.

Full Performs client certificate validation if the client presents a certificate.

Required
Requires the client to present a certificate and performs client certificate validation. This is the default.

SAFCheck
Requires the client to present a certificate, performs client certificate validation and requires the client certificate to have an associated user ID defined to the security product.

`SecondaryMap`
Specifies whether the application establishes secondary connections that should use the secondary policy mapping method. When specified in the

TTLSEnvironmentAdvancedParms, this statement overrides the value specified in the TTLSGroupAdvancedParms. Valid values are:

Off A connection that maps to this policy should not be used as a primary connection in the secondary policy mapping method.

On A connection that maps to this policy should be used as a primary connection in the secondary policy mapping method. Future connections established between the same two IP addresses by the same process that do not map to any policy or map to a policy with a lower priority are considered secondary connections. These secondary connections use the same policy mapped by the associated primary connection.

TTLSGroupAction statement

Use the TTLSGroupAction statement to specify parameters for a Language Environment process required to support secure connections. The TTLSGroupAction statement indicates whether a selected connection should use AT-TLS security. It can also specify the environment variables the Language Environment process should be initiated with.

Syntax

```
►►──TTLSGroupAction──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬──────────────────────────────────────────┤
   ├─┤ TTLSGroupAction Parameters ├─┤
   └─}──────────────────────────────┘
```

TTLSGroupAction Parameters:

```
                        ┌─CtraceClearText Off──────┐  ┌─Trace 2──┐
├──TTLSEnabled──┬─On──┬──┼──────────────────────────┼──┼──────────┼────────────►
                └─Off─┘  └─CtraceClearText──┬─On──┬─┘  └─Trace  n─┘
                                            └─Off─┘

                                        ┌─FIPS140 Off──────────┐
►──┬──────────────────────────────────┬──┼──────────────────────┼─────────────►
   ├─TTLSGroupAdvancedParms───────────┤  └─FIPS140──┬─On──┬────┘
   └─TTLSGroupAdvancedParmsRef name───┘             └─Off─┘

►──┬────────────────────┬──────────────────────────────────────────────────────┤
   └─GroupUserInstance n─┘
```

Parameters

name

A string 1 - 32 characters in length specifying the name of this TTLSGroupAction statement.

TTLSEnabled

Indicates the action that should be applied to connections using this TTLSGroupAction statement.

On AT-TLS security is active. Data might be encrypted, based on other policy statements.

Off AT-TLS security is not active. Data is sent in the clear.

CtraceClearText

Specifies whether application data traced using Ctrace or data trace is shown as unencrypted data. This parameter is applied only to connections that have active AT-TLS security on the connection. CtraceClearText can be specified on multiple actions referenced by a common TTLSRule statement. The value specified on the TTLSGroupAction statement can be overriden for particular AT-TLS environments by specifying it on the TTLSEnvironmentAction statement, or for particular connections by specifying it on the TTLSConnectionAction statement. Valid values are:

Off Application data is not traced as clear text. This is the default.

On Application data is traced as clear text.

Trace

Specifies the level of AT-TLS tracing. The valid values for *n* are in the range 0 - 255. The sum of the numbers associated with each level of tracing selected is the value that should be specified as *n*. If *n* is an odd number, errors are written to joblog and all other configured traces are sent to syslogd.

The trace parameter can be specified on multiple actions referenced by a common TTLSRule statement. The value specified on the TTLSGroupAction statement can be overriden for particular AT-TLS environments by specifying it on the TTLSEnvironmentAction statement or for particular connections by specifying it on the TTLSConnectionAction statement.

0 No tracing is enabled.

1 (Error)
Errors are traced to the TCP/IP joblog.

2 (Error)
Errors are traced to syslogd. This is the default. The messages are issued with syslogd priority code **err**.

4 (Info)
Tracing of instances when a connection is mapped to an AT-TLS rule and when a secure connection is successfully initiated is enabled. The messages are issued with syslogd priority code **info**.

8 (Event)
Tracing of major events is enabled. The messages are issued with syslogd priority code **debug**.

16 (Flow)
Tracing of system SSL calls is enabled. The messages are issued with syslogd priority code **debug**.

32 (Data)
Tracing of encrypted negotiation and headers is enabled. This traces the negotiation of secure sessions. The messages are issued with syslogd priority code **debug**.

64 Reserved.

128 Reserved.

255 All tracing is enabled.

TTLSGroupAdvancedParms
An inline specification of a TTLSGroupAdvancedParms statement.

TTLSGroupAdvancedParmsRef
The name of a globally defined TTLSGroupAdvancedParms statement.

FIPS140
Specifies whether FIPS 140 support is enabled for this group. Enabling FIPS 140 mode provides a higher degree of assurance of the integrity of the cryptographic modules that AT-TLS uses, including ICSF and System SSL. However, enabling FIPS 140 mode might require additional setup and configuration and it will restrict the available set of cryptographic algorithms. Valid values are:

Off Indicates that FIPS 140 is not supported for this group. This is the default.

On Indicates that FIPS 140 is supported for this group.

Requirement: ICSF must be active before starting AT-TLS groups configured to support FIPS140. For information about configuring ICSF to support FIPS 140-2, see Operating in compliance with FIPS 140-2 in z/OS Cryptographic Services ICSF Writing PKCS #11 Applications.

If the CSFSERV class is defined, give the userID that is associated with the TCPIP stack and any application userID using the TTLSGroup READ access to the CSFRNG resource within the RACF CSFSERV class. If the CSFSERV class is defined and Diffie Hellman is being used, give the application userID READ access to the CSF1TRC, CSF1DVK, CSF1GKP, CSF1GSK, CSF1GAV, and CSF1TRD resources within the RACF CSFSERV class.

GroupUserInstance
Defines a configurable instance identifier for this TTLSGroupAction statement. The *n* value can be in the range 0 - 65535. This parameter can be used to signal a change to the Policy Agent without modifying any of the other AT-TLS configuration statements. For example, when the contents of the Envfile has changed, but the Envfile file name is unchanged. Adding or updating the GroupUserInstance parameter would signal policy agent to install a new TTLSGroupAction statement. This parameter can also be used as a field to be updated when a change is made to this TTLSGroupAction statement. This enables the user to differentiate TTLSGroupAction statements, based on the instance identifier.

TTLSGroupAdvancedParms statement

Use the TTLSGroupAdvancedParms statement to specify advanced attributes for an AT-TLS group.

Syntax

```
►►──TTLSGroupAdvancedParms──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                            └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬─────────────────────────────────────────┤
   ├─┤ TTLSGroupAdvancedParms Parameters ├┤
   └─}──────────────────────────────────┘
```

TTLSGroupAdvancedParms Parameters:

```
   ┌─SecondaryMap Off─────────┐ ┌─SyslogFacility daemon────────┐
├──┴─SecondaryMap──┬─On──┬────┴─┴─SyslogFacility──┬─daemon─┬───┴──┬────────────┬──┤
                   └─Off─┘                        └─auth───┘      └─Envfile file─┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this TTLSGroupAdvancedParms statement.

  Rule: If this TTLSGroupAdvancedParms statement is not specified inline within another statement, a *name* value must be provided. If a name value is not specified for an inline TTLSGroupAdvancedParms statement a nonpersistent system name is created.

SecondaryMap
: Specifies whether the application establishes secondary connections that should use the secondary policy mapping method. Valid values are:

  Off A connection that maps to this policy should not be used as a primary connection in the secondary policy mapping method. This is the default.

  On A connection that maps to this policy should be used as a primary connection in the secondary policy mapping method. Future connections established between the same two IP addresses by the same process that do not map to any policy or map to a policy with a lower priority are considered secondary connections. These secondary connections use the same policy mapped by the associated primary connection.

SyslogFacility
: Specifies which syslog facility name this group should use when writing messages to syslogd. The daemon facility is currently used by other TCP/IP stack functions. See Chapter 20, "Syslog daemon," on page 909 for more information about these functions. Specifying auth enables syslog messages written by this AT-TLS group to be easily separated from messages written by other AT-TLS groups or other applications running in the same TCP/IP address space. Valid values are:

  daemon
  The daemon facility name is used. This is the default.

  auth The auth facility name is used.

Envfile
: Specifies the name of a file that contains environment variables. The Language Environment process is initialized with the _CEE_ENVFILE environment variable set to this file. See z/OS XL C/C++ Programming Guide for more information about the _CEE_ENVFILE environment variable.

The *file* value is a z/OS UNIX path and file name, a fully qualified MVS data set, specified as //fully.qualified.name, or a DD statement defined to the TCP/IP stack, specified as *DD:ddname*, containing environment variables. The maximum length is 1 023 characters. MVS data sets should be defined with variable-length records.

Restriction: The GSK_TRACE environment variable should not be set using the Envfile parameter. Setting this variable could cause unpredictable results or abends when running applications using AT-TLS. If System SSL trace data is needed, see z/OS Cryptographic Services System SSL Programming for information about running the SSL started task to gather trace data.

TTLSGskAdvancedParms statement

Use the TTLSGskAdvancedParms statement to specify advanced attributes for an AT-TLS environment that are specific to System SSL.

Syntax

```
►►──TTLSGskAdvancedParms──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                          └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────────┬─────────────────────────────────────────┤
   ├──┤ TTLSGskAdvancedParms Parameters ├──┤
   └─}──────────────────────────────────────┘
```

TTLSGskAdvancedParms Parameters:

```
├──┬─────────────────────────────┬──┬────────────────────────────────┬──────────────────►
   ├─TTLSGskLdapParms────────────┤  └─GSK_SYSPLEX_SIDCACHE──┬─On──┬─┘
   └─TTLSGskLdapParmsRef name────┘                          └─Off─┘

►──┬───────────────────────────────┬──┬─────────────────────────────┬──────────────────►
   └─GSK_V2_SESSION_TIMEOUT value──┘  └─GSK_V2_SIDCACHE_SIZE value──┘

►──┬───────────────────────────────┬──┬─────────────────────────────┬──────────────────┤
   └─GSK_V3_SESSION_TIMEOUT value──┘  └─GSK_V3_SIDCACHE_SIZE value──┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this TTLSGskAdvancedParms statement.

Rule: If this TTLSGskAdvancedParms statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inlineTTLSGskAdvancedParms statement, a nonpersistent system name is created.

TTLSGskLdapParms
An inline specification of a TTLSGskLdapParms statement.

TTLSGskLdapParmsRef
The name of a globally defined TTLSGskLdapParms statement.

GSK_SYSPLEX_SIDCACHE
Specifies whether sysplex session identifier caching is to be enabled for connections in this AT-TLS environment. Valid values are:

On Sysplex session identifier caching is to be enabled.

Off Sysplex session identifier caching is not to be enabled.

GSK_V2_SESSION_TIMEOUT
Specifies the SSL Version 2 session timeout. This is the number of seconds until a session identifier expires. Valid values are in the range 0 - 100.

GSK_V2_SIDCACHE_SIZE
Specifies the number of SSL Version 2 session identifiers to cache. Valid values are in the range 0 - 32 000.

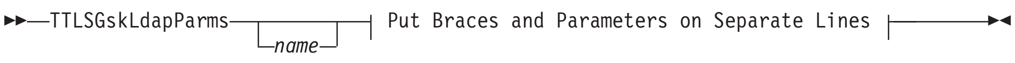
GSK_V3_SESSION_TIMEOUT
Specifies the SSL Version 3, TLS Version 1.0, or TLS Version 1.1 session timeout. This value is the number of seconds that lapse until a session identifier expires. Valid values are in the range 0 - 86 400.

GSK_V3_SIDCACHE_SIZE
Specifies the number of SSL Version 3, TLS version 1.0, or TLS version 1.1 session identifiers to cache. Valid values are in the range 0 - 64 000.

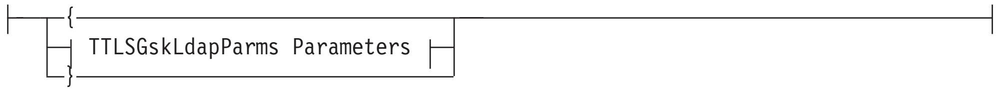
TTLSGskLdapParms statement

Use the TTLSGskLdapParms statement to define a set of LDAP parameters to be used for Certificate Revocation List (CRL) checking for an AT-TLS environment action. A TTLSGskLdapParms statement can be specified inline in a TTLSEnvironmentAction statement or referenced by an TTLSEnvironmentAction statement.

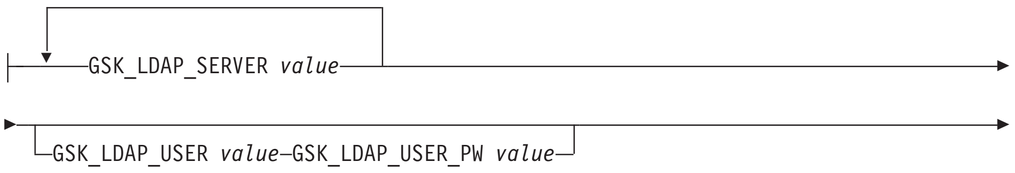
Syntax



Put Braces and Parameters on Separate Lines:



TTLSGskLdapParms Parameters:



```
►──┬──────────────────────────┬──┬────────────────────────────┬──────────────►
   └─GSK_LDAP_SERVER_PORT value─┘  └─GSK_CRL_CACHE_TIMEOUT value─┘

►──┬──────────────────────────────────────┬──────────────────────────────────┤
   └─GSK_CRL_SECURITY_LEVEL──┬─Low────┬──┘
                             ├─Medium─┤
                             └─High───┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this TTLSGskLdapParms statement.

  Rule: If this TTLSGskLdapParms statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inlineTTLSGskLdapParms statement, a nonpersistent system name is created.

GSK_LDAP_SERVER
: Specifies an LDAP server host name. The name can contain an optional port number separated from the name by a colon. The name can be a DNS resource name, a dotted-decimal IPv4 address or a colon-separated IPv6 address enclosed in square brackets (for example, [1080::8:800:200C:417A]). The maximum length of the host name is 255 characters. Valid values for the port number, if specified, are 1 - 65 535. Up to five GSK_LDAP_SERVER statements can be defined.

GSK_LDAP_USER
: Specifies the distinguished name to use when connecting to the LDAP server. The maximum length of the name is 512 characters.

  Rule: Comment indicators and embedded blanks are treated as part of the value for this attribute. For example:

```
GSK_LDAP_USER  cn=cert #label
value used:   cn=cert #label
```

  Restriction: When the value contains embedded blanks, you must specify the entire value within the first 1 536 characters of the configuration file line.

GSK_LDAP_USER_PW
: Specifies the password to use when connecting to the LDAP server. The maximum length of the password is 512 characters.

GSK_LDAP_SERVER_PORT
: Specifies the LDAP server port. This port is used if a port is not specified on the LDAP server host name. Valid values are in the range 1 - 65 535.

GSK_CRL_CACHE_TIMEOUT
: Sets the CRL cache timeout in hours. Valid values are in the range 0 - 720.

GSK_CRL_SECURITY_LEVEL
: Specifies the level of security to use when contacting an LDAP server. Valid values are:

  Low Specifies that certificate validation does not fail if the LDAP server cannot be contacted.

Medium
Specifies that certificate validation requires the LDAP server to be able to be contacted, but it does not require a CRL to be defined.

High Specifies that certificate validation requires the LDAP server to be contactable, and a CRL must be defined.

Tip: The located CRLs are cached according to the GSK_CRL_CACHE_TIMEOUT paramenter setting of the SSL environment.

TTLSKeyringParms statement

Use the TTLSKeyringParms statement to define a set of key ring parameters for an AT-TLS environment action. A TTLSKeyringParms statement can be specified inline in a TTLSEnvironmentAction statement or referenced by a TTLSEnvironmentAction statement.

Syntax

```
►►──TTLSKeyringParms──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                      └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├───┬─{──────────────────────────────────┬──────────────────────────────────────────────────┤
    ├──┤ TTLSKeyringParms Parameters ├──┤
    └─}──────────────────────────────────┘
```

TTLSKeyringParms Parameters:

```
├──Keyring value──┬─────────────────┬──┬───────────────────────┬───────────────────────────┤
                  └─KeyringPw value─┘  └─KeyringStashFile value─┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this TTLSKeyringParms statement.

Rule: If this TTLSKeyringParms statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline TTLSKeyringParms statement, a nonpersistent system name is created.

`Keyring`
Specifies the path and file name of the key database z/OS UNIX file, the ring name of the SAF key ring, or the name of the z/OS PKCS #11 token. For System SSL, a key database is assumed if KeyringPw or KeyringStashFile is also specified. For System SSL, a z/OS PKCS #11 token name is specified as `*TOKEN*/token-name`. `*TOKEN*` indicates that the specified key ring is actually a token name. Otherwise, a SAF key ring is used. For System SSL, the GSK_KEYRING_FILE value is set to the value specified. Valid values are 1 - 1 023 characters in length.

KeyringPw
: Specifies the password for the key database. For System SSL, GSK_KEYRING_PW is set to this value. Valid values are in the range 1 - 128 characters in length.

KeyringStashFile
: Specifies the path and file name of the key database password stash file. For System SSL, the KeyringPw value is used instead of the KeyringStashFile value, if that value is also specified. For System SSL, GSK_KEYRING_STASH_FILE is set to this value. Valid values are in the range 1 - 1 023 characters in length.

TTLSRule statement

Use the TTLSRule statement to define an AT-TLS rule.

The FLUSH/NOFLUSH and PURGE/NOPURGE parameters can be used to specify whether or not AT-TLS policies are deleted at startup (and when a MODIFY REFRESH command is entered) and shutdown, respectively.

The information provided on the TTLSRule statement defines an AT-TLS rule. The AT-TLS rule must have at least one local IP address, remote IP address, local port, remote port, job name, or user ID specification. The AT-TLS rule must have a direction specification and a TTLSGroupActionRef parameter. The AT-TLS rule can contain a priority, a TTLSConnectionActionRef parameter, a TTLSEnvironmentActionRef parameter and an IpTimeCondition specification. An IpTimeCondition specification identifies a time period when the AT-TLS rule is in effect.

Syntax

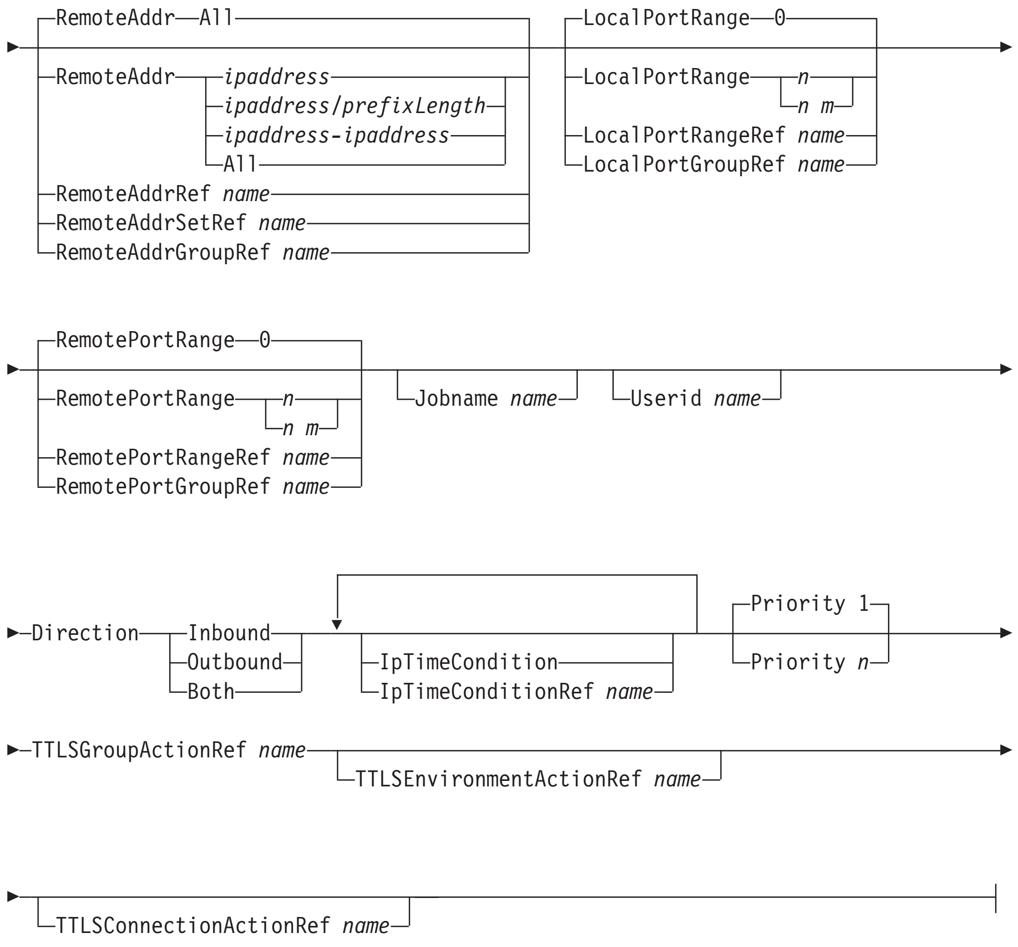
```
►►──TTLSRule──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────┬──────────────────────────────────────────────┤
   ├─┤ TTLSRule Parameters ├─┤
   └─}─────────────────────┘
```

TTLSRule Parameters:

```
   ┌─LocalAddr──All──────────────────────────┐
├──┼─────────────────────────────────────────┼─────────────────────────────►
   ├─LocalAddr──┬─ipaddress────────────────┬─┤
   │            ├─ipaddress/prefixLength───┤ │
   │            ├─ipaddress-ipaddress──────┤ │
   │            └─All──────────────────────┘ │
   ├─LocalAddrRef name───────────────────────┤
   ├─LocalAddrSetRef name────────────────────┤
   └─LocalAddrGroupRef name──────────────────┘
```



Parameters

name
A string 1 - 32 characters in length specifying the name of this TTLSRule statement.

LocalAddr
A local IP address the application is using for the connection that must match for this rule's action to be performed. The application can be explicitly bound to the IP address, or it can be chosen by the TCP/IP stack.

All Any local IP address matches this rule.

ipaddress
A single IP address.

ipaddress/prefixLength
The number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the unmasked bits defined.

ipaddress-ipaddress
A range of IP addresses.

Tip: To create a rule that matches only on local IPv4 addresses, code 0.0.0.0/0. To create a rule that matches only on local IPv6 addresses, code ::/0.

LocalAddrRef
The name of a globally defined IpAddr statement to be used for the local IP address specification.

LocalAddrSetRef
The name of a globally defined IpAddrSet statement to be used for the local IP address prefix or range specification.

LocalAddrGroupRef
The name of a globally defined IpAddrGroup statement to be used for the local IP address specification.

RemoteAddr
A remote IP address specification that must match for this rule's action to be performed.

All Any remote IP address matches this rule.

ipaddress
A single IP address.

ipaddress/prefixLength
The number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the ranges of 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP packet matches this condition if its unmasked bits are identical to the unmasked bits defined.

ipaddress-ipaddress
A range of IP addresses.

Tip: To create a rule that matches only on remote IPv4 addresses, code 0.0.0.0/0. To create a rule that matches only on remote IPv6 addresses, code ::/0.

RemoteAddrRef
The name of a globally defined IpAddr statement to be used for the remote IP address specification.

RemoteAddrSetRef
The name of a globally defined IpAddrSet statement to be used for the remote IP address prefix or range specification.

RemoteAddrGroupRef
The name of a globally defined IpAddrGroup statement to be used for the remote IP address specification.

LocalPortRange
A local port the application is bound to for this rule's action to be performed.

Valid values for *n* are in the range 0 - 65535. If 0 is specified for *n* then the rule applies to any local port. If *n* is specified as the beginning value for a range, then 0 is not a valid value.

If an *m* value is specified, it must be greater than or equal to *n* and less than 65536.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

LocalPortRangeRef
The name of a globally defined PortRange statement to be used for the local port specification.

LocalPortGroupRef
The name of a globally defined PortGroup statement to be used for the local port specification.

RemotePortRange
A remote port the application must be connecting to for this rule's action to be performed.

Valid values for *n* are in the range 0 - 65535. If 0 is specified for *n*, then the rule applies to any remote port. If *n* is specified as the beginning value for a range, then 0 is not a valid value.

If an *m* value is specified, then it must be greater than or equal to *n* and less than 65536.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

RemotePortRangeRef
The name of a globally defined PortRange statement to be used for the remote port specification.

RemotePortGroupRef
The name of a globally defined PortGroup statement to be used for the remote port specification.

Jobname
The *name* value specifies the job name of the application. This optional value specifies that, when the traffic is mapped to an AT-TLS security level, a packet must be flowing to or from an application with this job name for that packet to match the set of traffic characteristics. The *name* value must be 1 to 8 characters in length. It cannot include blanks or the "#" characters. A trailing asterisk indicates a wildcard specification. The specified job name is not case sensitive, and is translated to uppercase before being compared.

Userid
The *name* value specifies the corresponding user name. This optional value specifies that, when the traffic is mapped to an AT-TLS security level, a packet must be flowing to or from an application that is running under this user ID for that packet to match the set of traffic characteristics. The *name* value must be 1 to 8 characters in length. It cannot include blanks or the "#" characters. A trailing asterisk indicates a wildcard specification. The specified user ID is not case sensitive, and is translated to uppercase before being compared.

Direction
Specifies the direction the connection must be initiated from for this rule's action to be performed.

Inbound
A connection request has arrived inbound to the local host. An application must do an accept to service this connection.

Outbound
A connection request is being initiated by the local host. An application must have done a connect to initiate this connection.

Both Inbound and Outbound connection requests match this rule.

IpTimeCondition
An inline specification of a IpTimeCondition statement. There is a limit of 25 IpTimeCondition specifications on the TTLSRule statement.

IpTimeConditionRef
The name of a globally defined IpTimeCondition statement. There is a limit of 25 IpTimeCondition references on the TTLSRule statement.

Priority
An integer value in the range 1 - 2000000000 that represent the priority associated with the rule. The highest priority value is 2000000000.

Only one rule is ever mapped per connection. Rules are searched for a match starting at the highest priority, so if multiple rules could possibly be matched for a connection, the rule with the highest priority is matched first. If multiple rules of the same priority match, the rule that is mapped is difficult to predict. If this attribute is not specified, the default priority is 1.

Guideline: When setting the priority for multiple rules, do not set the priority as a sequential value, for example, 2, 3 ,4, 5. Instead, set the priority to provide space to change the priority or to insert additional rules, such that this rule is preferred over another rule, without duplicating a priority. For example, the priorities could be configured as 20, 30, 40, 50.

TTLSGroupActionRef
The name of a globally defined TTLSGroupAction statement.

TTLSEnvironmentActionRef
The name of a globally defined TTLSEnvironmentAction statement.

TTLSConnectionActionRef
The name of a globally defined TTLSConnectionAction statement.

Rules:

- One of the following values must be specified:
  - Local address
  - Remote address
  - Local port
  - Remote port
  - Job name
  - Userid
- A TTLSEnvironmentActionRef is required if the TTLSGroupAction specifies TTLSEnabled as **On**.
- CNF logic is used to evaluate complex AT-TLS rules (rules containing multiple conditions). For a detailed description of AT-TLS condition evaluation using CNF logic, see z/OS Communications Server: IP Configuration Guide. An AT-TLS condition is comprised of the following values from the TTLSRule statement:
  - Local IP Address
  - Remote IP Address
  - Local Port
  - Remote Port
  - Jobname
  - Userid
  - Service Direction

TTLSSignatureParms statement

Use the TTLSSignatureParms statement to define the client elliptic curve preferences and the signature algorithm pair specifications for an AT-TLS environment or an AT-TLS connection. A TTLSSignatureParms statement can be specified inline in a TTLSEnvironmentAction or TTLSConnectionAction statement or referenced by a TTLSEnvironmentAction or TTLSConnectionAction statement.

Syntax

```
►►──TTLSSignatureParms──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                        └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{───────────────────────────────┬──────────────────────────────────────────────────────┤
   ├─┤ TTLSSignatureParms Parameters ├─┤
   └─}─────────────────────────────────┘
```

TTLSSignatureParms Parameters:

```
├──┬────────────────────────────────────┬──┬───────────────────────────────────┬──────────────┤
   ├─ClientECurves──Any─────────────────┤  └─SignaturePairsParms──parameters───┘
   └─ClientECurveParms── parameters─────┘
```

ClientECurveParms Parameters:

```
     ┌────────────────────────────┐
     ▼                            │
├────────ClientECurves──curves────┴─────────────────────────────────────────────────────────┤
```

SignaturePairsParms Parameters:

```
     ┌────────────────────────────────────┐
     ▼                                    │
├────────SignaturePairs──algorithms───────┴─────────────────────────────────────────────────┤
```

Parameters

name
: A string 1 - 32 characters in length that specifies the name of this TTLSSignatureParms statement.

  Rule: If this TTLSSignatureParms statement is not specified inline in another statement, a *name* value must be provided. If a name is not specified for an inline TTLSSignatureParms statement, a nonpersistent system name is created.

ClientECurves
: Specifies the list of elliptic curves that are supported by the client, in order of preference for use. The elliptical curve specifications are used by the client to tell the server which elliptical curves can be used when using cipher suites that use elliptical curve cryptography for the TLSv1.0 protocol or later.

Only curves that are recommended by NIST can be specified. To allow the use of Brainpool standard curves in addition to NIST standard curves for an SSL connection, the list must contain only the ANY curve name constant. If a ClientECurves parameter is specified more than once, the values are concatenated to create a single list of elliptic curve enumerators. The ANY curve name constant cannot be specified in combination with any NIST curves. For System SSL, the GSK_CLIENT_ECURVE_LIST value is set to the concatenated value or to NULL if ANY is specified.

The *curves* value is a string of one or more 4-character curve enumerators or a single curve name constant. The curve string cannot have blanks between the curve enumerators. If duplicate curves are specified, the first instance is used and all other instances are ignored. The maximum number of curves is 16. For System SSL, see Table 16. Supported elliptic curve definitions for TLS V1.0, TLS V1.1 and TLS V1.2 in z/OS Cryptographic Services System SSL Programming for a list of valid elliptic curves. Table 71 lists the supported elliptic curve name constants.

Table 71. ClientEcurves

Elliptic curve name constants	Elliptic Curve Enumerator
secp192r1	0019
secp224r1	0021
secp256r1	0023
secp384r1	0024
secp521r1	0025

Requirement: Elliptic Curve requires ICSF to be active. See Elliptic Curve Cryptography Support in z/OS Cryptographic Services System SSL Programming for more information.

SignaturePairs

Specifies the TLS version 1.2 signature algorithm pairs supported for the server certificate. These pairs are sent by the client when proposing use of the TLSv1.2 protocol to indicate to the server which signature/hash algorithm pairs may be used in digital signatures of the server certificate. SignaturePairs is only meaningful when performing a handshake with a Server that supports the TLSv1.2 protocol and will be ignored by any Server that only supports TLSv1.1 or earlier.

If a SignaturePairs parameter is specified more than once, the values are concatenated to create a single list of signature algorithm pairs. For System SSL, the GSK_TLS_SIG_ALG_PAIRS value is set to the concatenated value. If not specified, then System SSL will use a default list of acceptable signature algorithm pairs.

The *algorithms* value is a string of one or more 4-character TLS version 1.2 signature algorithm pairs or a single signature algorithm pair constant. The algorithm string cannot have blanks between each TLS version 1.2 signature algorithm pair. If duplicate signature algorithm pairs are specified, the first instance is used and all other instances are ignored. The maximum number of TLS version 1.2 signature algorithm pairs is 64. For System SSL, see Table 17. Signature Algorithm pair definitions for TLS V1.2 in z/OS Cryptographic Services System SSL Programming for a list of valid signature algorithm pairs. Table 72 on page 1043 lists the supported signature algorithm pair constants.

Table 72. SignaturePairs

Signature algorithm pair constant	Hexadecimal characters
TLS_SIGALG_MD5_WITH_RSA	0101
TLS_SIGALG_SHA1_WITH_RSA	0201
TLS_SIGALG_SHA1_WITH_DSA	0202
TLS_SIGALG_SHA1_WITH_ECDSA	0203
TLS_SIGALG_SHA224_WITH_RSA	0301
TLS_SIGALG_SHA224_WITH_DSA	0302
TLS_SIGALG_SHA224_WITH_ECDSA	0303
TLS_SIGALG_SHA256_WITH_RSA	0401
TLS_SIGALG_SHA256_WITH_DSA	0402
TLS_SIGALG_SHA256_WITH_ECDSA	0403
TLS_SIGALG_SHA384_WITH_RSA	0501
TLS_SIGALG_SHA384_WITH_ECDSA	0503
TLS_SIGALG_SHA512_WITH_RSA	0601
TLS_SIGALG_SHA512_WITH_ECDSA	0603

IDS policy statements

This topic contains information about the following IDS policy statements:
- “IDSAction statement”
- “IDSAttackCondition statement” on page 1046
- “IDSExclusion statement” on page 1055
- “IDSReportSet statement” on page 1057
- “IDSRule statement” on page 1060
- “IDSScanEventCondition statement” on page 1063
- “IDSScanExclusion statement” on page 1065
- “IDSScanGlobalCondition statement” on page 1067
- “IDSTRCondition statement” on page 1068

IDSAction statement

Use the IDSAction statement to define the action taken by the IDS rule. This statement is associated with an IDS rule with the same ActionType value.

```
►►──IDSAction──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IDSAction Parameters ├─┤
   └─}──────────────────────┘
```

IDSAction Parameters:

```
   ┌──────────────────────────────────────────┐
   ▼                                          │
├───ActionType──┬─Attack──┬─discard──────┬──┬──┤ ReportSet ├──────────────┤
                │         ├─nodiscard────┤  │
                │         ├─resetconn────┤  │
                │         └─noresetconn──┘  │
                ├─TR──┬─limit───┬───────────┤
                │     └─nolimit─┘           │
                ├─ScanGlobal────────────────┤
                └─ScanEvent──count──────────┘
```

ReportSet:

```
├──┬──────────────────────────┬──────────────────────────────────────────┤
   ├─IDSReportSet─────────────┤
   └─IDSReportSetRef name─────┘
```

Parameters

name
A string 1 - 32 characters in length that specifies the name of this IDSAction statement.

ActionType
Indicates the type of IDS action associated with a policy rule.

Attack Indicates that this is an attack action.

Discard
Discard packets that match the associated rule.

NoDiscard
Do not discard packets that match the associated rule.

ResetConn
Reset the TCP connection or connections associated with the attack.

Restriction: This value is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

NoResetConn
Do not reset the TCP connection or connections associated with the attack.

Restriction: This value is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

Rules:

- The Discard and NoDiscard values are valid for the following attack types:
  - DATA_HIDING
  - ICMP_REDIRECT
  - IP_FRAGMENT
  - OUTBOUND_RAW

- OUTBOUND_RAW_IPV6
- PERPETUAL_ECHO
- RESTRICTED_IP_OPTIONS
- RESTRICTED_IP_PROTOCOL
- RESTRICTED_IPV6_DST_OPTIONS
- RESTRICTED_IPV6_HOP_OPTIONS
- RESTRICTED_IPV6_NEXT_HDR
- FLOOD (NoDiscard is ignored because the stack always discards packets associated with a flood.)
- MALFORMED_PACKET (NoDiscard is ignored because the stack always discards malformed packets.)
- EE_MALFORMED_PACKET
- EE_PORT_CHECK
- EE_LDLC_CHECK
- EE_XID_FLOOD (The Discard value is not valid. Use the NoDiscard value.)

The Discard and NoDiscard values are ignored for all other attack types. For more information, see “IDSAttackCondition statement” on page 1046.

- The ResetConn and NoResetConn values are valid for the following attack types:
  - TCP_QUEUE_SIZE
  - GLOBAL_TCP_STALL

  ResetConn and NoResetConn will be ignored for all other attack types. For more information, see “IDSAttackCondition statement” on page 1046.
- An IDSAction statement can include two ActionType attack parameters, one with the action Discard or NoDiscard and the other with the action ResetConn or NoResetConn. If more than one ActionType attack parameter is coded with the action Discard or NoDiscard, the last action is used. If more than one ActionType attack parameter is coded with the action ResetConn or NoResetConn, the last action is used.

ScanGlobal
Indicates that this is a scan global action that specifies global scan detection values.

ScanEvent count
Indicates that this is a scan event action for individual scan detection.

count Increment the scan event counter for this rule.

TR Indicates that this is a traffic regulation action.

Limit For TCP, this value prevents connections, for UDP, it limits the length of inbound UDP queues.

NoLimit
No limits are placed on the number of TCP connections or the length of inbound UDP queues.

Rule: If you specify more than one ActionType TR parameter, the setting from the last parameter that you specified is used.

IDSReportSet
: An inline specification of an IDSReportSet statement.

IDSReportSetRef *name*
: The name of a globally defined IDSReportSet statement.

Rules:

- The IDSReportSet parameter is allowed for all ActionType values. However, it has no effect if specified for ActionType ScanEvent.
- Not all parameters specified on the IDSReportSet statement apply to all ActionType values. Such values are ignored by the stack when not applicable to the IDS policy.

IDSAttackCondition statement

Use the IDSAttackCondition statement for attack detection, reporting, and prevention. There are several attack types. For each attack type, the single highest priority rule is used.

The IDSAttackCondition statement can specify values for LocalPortRange, RemotePortRange, or both, or these values can be specified with references to global definitions on the PortRange or PortGroup statements.

The IDSAttackCondition statement can specify values for ProtocolRange, or this value can be specified with a reference to global definitions on the IPProtocolRange or IPProtocolGroup statements.

The IDSAttackCondition statement can specify values for the RestrictedIpOptionRange parameter, or this value can be specified with a reference to global definitions on the IpOptionRange or IpOptionGroup statements.

The IDSAttackCondition statement can specify values for the IPv6NextHdrRange parameter, or this value can be specified with a reference to global definitions on the IPv6NextHdrRange or IPv6NextHdrGroup statements.

The IDSAttackCondition statement can specify values for the RestrictedIpv6OptionRange parameter, or this value can be specified with a reference to global definitions on the IpOptionRange or IpOptionGroup statements.

The IDSAttackCondition statement can contain an inline definition of an IDSExclusion, or this value can be specified with a reference to a global definition of an IDSExclusion statement.

Syntax

```
►►──IDSAttackCondition──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                        └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{────────────────────────────────┬──────────────────────────────────────────────┤
   ├──┤ IDSAttackCondition Parameters ├──┤
   └─}────────────────────────────────┘
```

IDSAttackCondition Parameters:

```
├──AttackType──┬─DATA_HIDING──┤ DataHidingCond ├──────────────────────────┬──┤
               ├─EE_LDLC_CHECK──┤ IDSExclusionCond ├──────────────────────┤
               ├─EE_MALFORMED_PACKET──┤ IDSExclusionCond ├────────────────┤
               ├─EE_PORT_CHECK──┤ IDSExclusionCond ├──────────────────────┤
               ├─EE_XID_FLOOD──┤ EEXIDFloodCond ├─────────────────────────┤
               ├─FLOOD──┤ FloodCond ├─────────────────────────────────────┤
               ├─GLOBAL_TCP_STALL─────────────────────────────────────────┤
               ├─ICMP_REDIRECT────────────────────────────────────────────┤
               ├─IP_FRAGMENT──────────────────────────────────────────────┤
               ├─MALFORMED_PACKET─────────────────────────────────────────┤
               ├─OUTBOUND_RAW──┤ IpProtocolCond ├─────────────────────────┤
               ├─OUTBOUND_RAW_IPv6──┤ IpProtocolCond ├────────────────────┤
               ├─PERPETUAL_ECHO──┤ PerpetualEchoCond ├────────────────────┤
               ├─RESTRICTED_IP_OPTIONS──┤ RestrictedIpOptionsCond ├───────┤
               ├─RESTRICTED_IP_PROTOCOL──┤ IpProtocolCond ├───────────────┤
               ├─RESTRICTED_IPV6_DST_OPTIONS──┤ RestrictedIPv6OptionsCond ├─┤
               ├─RESTRICTED_IPV6_HOP_OPTIONS──┤ RestrictedIPv6OptionsCond ├─┤
               ├─RESTRICTED_IPV6_NEXT_HDR──┤ RestrictedIPv6NextHdrCond ├──┤
               └─TCP_QUEUE_SIZE──┤ TcpQueueSizeCond ├─────────────────────┘
```

DataHidingCond:

```
   ┌─OptionPadChk Enable──────────┐ ┌─IcmpEmbedPktChk Enable──────────┐
├──┴─OptionPadChk ──┬─Disable─┬──┴─┴─IcmpEmbedPktChk ──┬─Disable─┬──┴──────┤
                    └─Enable──┘                        └─Enable──┘
```

EEXIDFloodCond:

```
   ┌─EEXIDTimeout 100─┐
├──┴─EEXIDTimeout n───┴──┬─────────────────────────┬──────────────────────┤
                         ├─IDSExclusion ───────────┤
                         └─IDSExclusionRef name────┘
```

FloodCond:

```
   ┌─IfcFloodMinDiscard 1000─┐ ┌─IfcFloodPercentage 10─┐
├──┴─IfcFloodMinDiscard n────┴─┴─IfcFloodPercentage n──┴─────────────────┤
```

IDSExclusionCond:

```
├──┬─────────────────────────┬─────────────────────────────────────────────┤
   ├─IDSExclusion ───────────┤
   └─IDSExclusionRef name────┘
```

IpProtocolCond:

```
├──┬─ProtocolRange──┬─n───┬─┬──────────────────────────────────────────────────┤
   │                └─n m─┘ │
   ├─ProtocolRangeRef name──┤
   └─ProtocolGroupRef name──┘
```

PerpetualEchoCond:

```
├──┬─LocalPortRange──┬─n───┬─┬──┬─RemotePortRange──┬─n───┬─┬────────────────────┤
   │                 └─n m─┘ │  │                  └─n m─┘ │
   ├─LocalPortRangeRef name──┤  ├─RemotePortRangeRef name──┤
   └─LocalPortGroupRef name──┘  └─RemotePortGroupRef name──┘
```

RestrictedIPOptionsCond:

```
   ┌─RestrictedIpOptionRange All──────┐
├──┼──────────────────────────────────┼───────────────────────────────────────────┤
   ├─RestrictedIpOptionRange──┬─n───┬─┤
   │                          ├─n m─┤ │
   │                          └─All─┘ │
   ├─RestrictedIpOptionRangeRef name──┤
   └─RestrictedIpOptionGroupRef name──┘
```

RestrictedIPv6OptionsCond:

```
├──┬─RestrictedIpv6OptionRange──┬─n───┬─┬───────────────────────────────────────────┤
   │                            └─n m─┘ │
   ├─RestrictedIpv6OptionRangeRef name──┤
   └─RestrictedIpv6OptionGroupRef name──┘
```

RestrictedIPv6NextHdrCond:

```
├──┬─IPv6NextHdrRange──┬─n───┬─┬──────────────────────────────────────────────────┤
   │                   └─n m─┘ │
   ├─IPv6NextHdrRangeRef name──┤
   └─IPv6NextHdrGroupRef name──┘
```

TcpQueueSizeCond:

```
   ┌─TcpQueueSize SHORT───────────┐
├──┴─TcpQueueSize ──┬─LONG───────┬┴──┬─────────────────────────┬──────────────────┤
                    ├─VERY_LONG──┤   ├─IDSExclusion ───────────┤
                    ├─VERY_SHORT─┤   └─IDSExclusionRef name────┘
                    └─SHORT──────┘
```

Parameters

name
: A string 1 - 32 characters in length that specifies the name of this IDSAttackCondition statement.

Rule: If this IDSAttackCondition statement is not specified inline within another statement, *name* must be provided. If a name is not specified for an inline IDSAttackCondition statement, a nonpersistent system name is created.

`AttackType`

DATA_HIDING
Indicates that the rule is for detecting hidden data. The DATA_HIDING attack type applies to both IPv4 and IPv6 inbound packets.

Restriction: This value is valid only for V1R13 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

EE_MALFORMED_PACKET
Indicates that the rule is for EE malformed packets. The packets can be discarded by TCP/IP or forwarded to VTAM. The EE_MALFORMED_PACKET attack type applies to both IPv4 and IPv6 malformed packets.

Restriction: This value is valid only for V1R13 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

EE_PORT_CHECK
Indicates that the rule checks the source port number for inbound Enterprise Extender (EE) packets. The source port number must be the same as the destination port number. The EE_PORT_CHECK attack type applies to both IPv4 and IPv6 packets.

Restriction: This value is valid only for V1R13 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

EE_LDLC_CHECK
Indicates that the rule is for LDLC control commands received on a port other than the signalling port. The EE_LDLC_CHECK attack type applies to both IPv4 and IPv6 packets.

Restriction: This value is valid only for V1R13 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

EE_XID_FLOOD
Indicates that the rule is for an EE XID flood attack. The EE_XID_FLOOD attack type applies to both IPv4 and IPv6.

Restriction: This value is valid only for V1R13 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

FLOOD
Indicates that the rule is for flooding attacks. For FLOOD attacks, the packets are always discarded regardless of what ActionType is configured on the IDSAction. The FLOOD attack type applies to both IPv4 and IPv6.

GLOBAL_TCP_STALL
Indicates that the rule is to detect an attack that causes a large number of TCP connections to be stalled and unable to send data. The GLOBAL_TCP_STALL attack type applies to both IPv4 and IPv6 connections.

Results:

- A global TCP stall condition is detected for a TCP/IP stack when at least 50% of the active TCP connections are stalled and at least 1000 TCP connections are active.

- When the condition is detected, the stalled TCP connections are reset if the policy action specifies resetconn.
- When the condition is detected, a syslogd message is generated for each stalled connection if TypeActions Log LogDetail is specified. Message EZZ8673I is generated if the stalled connection is reset. Otherwise, message EZZ8674I is generated.

Restriction: This value is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

ICMP_REDIRECT
Indicates that the rule is for ICMP redirect detection. This includes both ICMP redirects and ICMPv6 redirects.

IP_FRAGMENT
Indicates that the rule is for detecting suspicious fragmented packets (fragments that overlay and change data in the packet, including changes to the length of the packet).

MALFORMED_PACKET
Indicates that the rule is for a number of specific malformed packets that are detected on inbound traffic. For MALFORMED_PACKET attacks, the packets are always discarded regardless of what ActionType is configured on the IDSAction. The MALFORMED_PACKET attack type applies to both IPv4 and IPv6 inbound packets.

OUTBOUND_RAW
Indicates that the rule is to enforce restrictions on the use of IPv4 RAW sockets for outbound processing, which prevents this stack from being used to attack other systems. A list of restricted IP protocols is also specified in the rule's conditions.

Restriction: The OUTBOUND_RAW attack type applies only to IPv4 packets. The OUTBOUND_RAW_IPV6 attack type provides analogous function for IPv6 packets.

OUTBOUND_RAW_IPV6
Indicates that the rule is to enforce restrictions on the use of IPv6 RAW sockets for outbound processing, which prevents this stack from being used to attack other systems. A list of restricted protocols is also specified in the rule's conditions.

Rule: IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Restrictions:

- The OUTBOUND_RAW_IPV6 attack type applies only to IPv6 packets. The OUTBOUND_RAW attack type provides analogous function for IPv4 packets.
- This value is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

PERPETUAL_ECHO
Indicates that the rule is to prevent perpetual echos over UDP ports. A list of local UDP ports that always respond to an input packet is also specified in the rule's conditions, and a separate list of remote (network) UDP ports that always respond is specified. The PERPETUAL_ECHO attack type applies to both IPv4 and IPv6 packets.

Rule: For PERPETUAL_ECHO attacks, only the first 20 ports specified in the local list and in the remote list are used.

RESTRICTED_IP_OPTIONS
Indicates that the rule is to detect inbound IPv4 packets that have IP options that are not allowed. A list of restricted IP options is also specified in the rule's conditions.

For RESTRICTED_IP_OPTIONS attacks, if no option ranges are specified, all options are restricted. Option 0 (end of option list) and 1 (no-operation) are always allowed; they are ignored if present in the list of restricted IP options.

Restriction: The RESTRICTED_IP_OPTIONS attack type applies only to IPv4 packets. The RESTRICTED_IPV6_NEXT_HDR, RESTRICTED_IPV6_DST_OPTIONS, and RESTRICTED_IPV6_HOP_OPTIONS attack types provide analogous function for IPv6 packets.

RESTRICTED_IP_PROTOCOL
Indicates that the rule is to detect inbound IPv4 packets that have IP protocols that are not allowed. A list of restricted IP protocols is also specified in the rule's conditions.

For RESTRICTED_IP_PROTOCOL attacks, Protocol 1 (ICMP), 6 (TCP), and 17 (UDP) are ignored if present in the list of restricted IP protocols..

Restriction: The RESTRICTED_IP_PROTOCOLS attack type applies only to IPv4 packets. The RESTRICTED_IPV6_NEXT_HDR attack type provides analogous function for IPv6 packets.

RESTRICTED_IPV6_DST_OPTIONS
Indicates that the rule is to detect inbound IPv6 packets that have an IPv6 destination options extension header with options that are not allowed. A list of restricted IPv6 destination option values is specified in the rule's conditions.

Rule: IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Restrictions:

- The RESTRICTED_IPV6_DST_OPTIONS attack type applies only to IPv6 packets. The RESTRICTED_IP_OPTIONS attack type provides analogous function for IPv4 packets.
- You cannot restrict options 0 (Pad1) or 1 (PadN).
- This value is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

RESTRICTED_IPV6_HOP_OPTIONS
Indicates that the rule is to detect inbound IPv6 packets that have an IPv6 hop-by-hop options extension header with options that are not allowed. A list of restricted IPv6 hop-by-hop option values is specified in the rule's conditions.

Rule: IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Restrictions:

- The RESTRICTED_IPV6_HOP_OPTIONS attack type applies only to IPv6 packets. The RESTRICTED_IP_OPTIONS attack type provides analogous function for IPv4 packets.
- You cannot restrict options 0 (Pad1) or 1 (PadN).
- This value is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

RESTRICTED_IPV6_NEXT_HDR

Indicates that the rule is to detect inbound IPv6 packets that have a next header value that is not allowed. A list of restricted IPv6 next header values is specified in the rule's conditions. The IPv6 packet header and any subsequent extension headers include a next header field that will be checked. The value in the next header field identifies the next header in the packet, either an upper layer protocol header (such as a TCP or UDP header) or an extension header (such as a fragmentation or routing header).

Rule: IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Restrictions:

- The RESTRICTED_IPV6_NEXT_HDR attack type applies only to IPv6 packets. The RESTRICTED_IP_OPTIONS and RESTRICTED_IP_PROTOCOL attack types provide analogous function for IPv4 packets.
- You cannot restrict next header values 6 (TCP), 17 (UDP), or 58 (ICMPv6).
- This value is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

TCP_QUEUE_SIZE

Indicates that the rule is to detect TCP send, receive, and out-of-order queues that are constrained. A queue can be constrained due to the amount of data on the queue or the age of the data on the queue. A queue size is specified in the rule's conditions. An exclusion list can optionally be specified in the rule's conditions. The TCP_QUEUE_SIZE attack type applies to both IPv4 and IPv6 connections.

Restriction: This value is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

OptionPadChk

Indicates whether checking for non-zero IP option pad fields in inbound packets should be enabled or disabled. The default is Enable. For IPv4 packets, the options field is in the IP header and can contain zero filled padding for alignment purposes. For IPv6 packets, a hop-by-hop options extension header or a destination options extension header can include one or more zero filled padding options for alignment purposes.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IcmpEmbedPktChk

Indicates whether checking of embedded packets within an inbound ICMP or ICMPv6 error message should be enabled or disabled. The default is Enable.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

EEXIDTimeout
Indicates the number of XID exchange timeouts that must occur within a 1-minute period in order to be detected as an EE XID flood attack. Valid values are in the range 1- 2 000 000 000. The default value is 100.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IfcFloodMinDiscard
Indicates the minimum number of discarded packets that must occur on an interface within a 1 minute period in order to be recognized as an interface flood attack. Valid values are in the range 100 - 4294967295. The default value is 1000.

IfcFloodPercentage
Indicates the percentage of discarded packets for an interface above which an interface flood attack is recognized. Valid values are in the range 5 - 100. The default value is 10.

ProtocolRange
Indicates the restricted protocols for this IDS attack rule.

n m Integers that specify a protocol range. Valid values for *n* are in the range 0 - 255. If an *m* value is specified, then it must be greater than or equal to *n* and less than 256.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

ProtocolRangeRef
The name of a globally defined IpProtocolRange statement.

ProtocolGroupRef
The name of a globally defined IpProtocolGroup statement.

RestrictedIpOptionRange
Indicates the restricted IPv4 options for this IDS attack rule.

All IP options 2 through 255 are restricted. Option 0 (end of option list) and 1 (no-operation) are always allowed and cannot be restricted by policy. This is the default value.

n m Integers that specify a restricted IP option range. Valid values for *n* are in the range 1 - 255. If an *m* value is specified, then it must be greater than or equal to *n* and less than 256.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

RestrictedIpOptionRangeRef
The name of a globally defined IpOptionRange statement.

RestrictedIpOptionGroupRef
The name of a globally defined IpOptionGroup statement.

RestrictedIpv6OptionRange
Indicates the restricted IPv6 options for this IDS attack rule.

n m Integers that specify a restricted option range. Valid values for *n* are in the range 2 - 255. If an *m* value is specified, then it must be greater than or equal to *n* and less than 256.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

RestrictedIpv6OptionRangeRef
The name of a globally defined IpOptionRange statement.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

RestrictedIpv6OptionGroupRef
The name of a globally defined IpOptionGroup statement.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IPv6NextHdrRange
Indicates the restricted IPv6 next header values for this IDS attack rule. The value in the next header field of an IPv6 header or extension header identifies the next header in the packet, either an upper layer protocol (such as TCP or UDP) or an extension header (such as fragmentation or routing).

n m Integers that specify a restricted IPv6 next header value range. Valid values for *n* are in the range 0 - 255. If an *m* value is specified, then it must be greater than or equal to *n* and less than 256.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

Restrictions:

- You cannot restrict next header values 6 (TCP), 17 (UDP), or 58 (ICMPv6). They are always allowed.
- This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IPv6NextHdrRangeRef
The name of a globally defined IPv6NextHdrRange statement.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IPv6NextHdrGroupRef
The name of a globally defined IPv6NextHdrGroup statement.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

LocalPortRange
A list of local ports for this IDS attack rule. Valid values for *n* are in the range 1 - 65535. If an *m* value is specified, then it must be greater than or equal to *n* and less than 65536.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

Restriction: A LocalPortRange or RemotePortRange of 0 is not allowed.

LocalPortRangeRef
The name of a globally defined PortRange statement to be used for the local port specification.

LocalPortGroupRef
The name of a globally defined PortGroup statement to be used for the local port specification.

RemotePortRange
A list of remote ports for this IDS attack rule. Valid values for *n* are in the range 1 - 65535. If an *m* value is specified then it must be greater than or equal to *n* and less than 65536.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

Restriction: A LocalPortRange or RemotePortRange of 0 is not allowed.

RemotePortRangeRef
The name of a globally defined PortRange statement to be used for the remote port specification.

RemotePortGroupRef
The name of a globally defined PortGroup statement to be used for the remote port specification.

TcpQueueSize
Indicates the amount of data that must remain on a TCP send, receive, or out-of-order queue for at least thirty seconds before the queue will become constrained. Note that a queue will also become constrained if any amount of data remains on the queue for at least sixty seconds. This parameter is used to select one of a number of abstract queue sizes that map to internally defined limits. For details about queue sizes, see the Attack policies information in z/OS Communications Server: IP Configuration Guide.

- VERY_SHORT
- SHORT (this is the default)
- LONG
- VERY_LONG

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IDSExclusion
An inline specification of an IDSExclusion statement.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IDSExclusionRef
The name of a globally defined IDSExclusion statement.

Restriction: This parameter is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IDSExclusion statement

Use the IDSExclusion statement to specify IP addresses, and optionally ports, that are to be excluded when monitoring for certain attacks. For example, you can use an IDSExclusion statement to exclude a printer connection from being reset by TCP_QUEUE_SIZE attack detection if the printer remains in a persist state for a period of time while out of paper. You can also use the IDSExclusion statement to exclude a host from the EE_PORT_CHECK attack detection if you know that the host uses ephemeral source ports for Enterprise Extender (EE) traffic.

Restriction: This statement is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

Syntax

```
►►──IDSExclusion──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                  └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{────────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IDSExclusion Parameters ├─┤
   └─}────────────────────────────┘
```

IDSExclusion Parameters:

```
    ┌─────────────────────────────────────────────────┐
    ▼                                                 │
├───ExcludedAddrPort──┬─ipaddress─────────────────────┬┴──────────────────────────┤
                      ├─ipaddress n───────────────────┤
                      ├─ipaddress n m─────────────────┤
                      ├─ipaddress/prefixLength────────┤
                      ├─ipaddress/prefixLength n──────┤
                      └─ipaddress/prefixLength n m────┘
```

Parameters

name

A string 1 - 32 characters in length that specifies the name of this IDSExclusion statement.

Rule: If you do not specify this IDSExclusion statement inline within another statement, you must provide a *name* value. If you do not specify a name for an inline IDSExclusion statement, a nonpersistent system name is created.

ExcludedAddrPort

Indicates the IP addresses, and optionally ports, that are to be excluded.

ipaddress

A single IPv4 or IPv6 address.

ipaddress n

A single IPv4 or IPv6 address and port. Valid port values for *n* are 0 - 65535. If you specify 0 for *n*, then all ports are excluded.

ipaddress n m

A single IPv4 or IPv6 address and range of ports. Valid port values for *n* are 1 - 65535. The *m* value must be greater than or equal to *n* and less than 65536.

ipaddress/prefixLength

An IPv4 or IPv6 prefix address specification. Valid values for the *prefixLength* value are in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the defined unmasked bits.

ipaddress/prefixLength n

An IPv4 or IPv6 prefix address specification and a port. Valid values for the *prefixLength* value are in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the defined unmasked bits. Valid port values for *n* are 0 - 65535. If you specify 0 for *n*, all ports are excluded.

ipaddress/prefixLength n m

An IPv4 or IPv6 prefix address specification and a range of ports. Valid values for the *prefixLength* value are in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the defined unmasked bits. Valid port values for *n* are 1 - 65535. The *m* value must be greater than or equal to *n* and less than 65536.

Result: Only the first 10,000 ExcludedAddrPort entries are saved and used.

Rule: You can specify a mix of IPv4 and IPv6 addresses in one IDSExclusion statement.

IDSReportSet statement

Use the IDSReportSet statement to specify a report set that you want to associate with actions. A report set can include type of action, statistics interval, logging level, trace data, and trace record size. If a packet meets a policy rule's condition during its validity period, the reports specified in the policy rule's action, such as logging the packet, are produced.

Syntax

```
►►──IDSReportSet──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                  └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬──────────────────────────────────────────────────┤
   ├──┤ IDSReportSet Parameters ├┤
   └─}──────────────────────────┘
```

IDSReportSet Parameters:

```
     ┌──────────────────────────────────────────────────────┐
     ▼                                                      │
├────┬──────────────────────────────────────────────────────┴──────────────────────┤
     └─TypeActions──┬─CONSOLE──┤ ConsoleData ├──────────┤
                    ├─LOG──┤ LogData ├──────────────────┤
                    ├─STATISTICS──┤ StatsData ├─────────┤
                    └─TRACE──┤ TraceData ├──────────────┘
```

ConsoleData:

```
   ┌─MaxEventMessage 5─┐
├──┴─MaxEventMessage n─┴──────────────────────────────────────────────────────────┤
```

LogData:

```
   ┌─LogDetail No──────────┐ ┌─LoggingLevel 4─┐
├──┴─LogDetail──┬─Yes─┬────┴─┴─LoggingLevel n─┴──────────────────────────────────┤
                └─No──┘
```

StatsData:

```
   ┌─StatType Normal───────────┐  ┌─StatInterval 60─┐
├──┼───────────────────────────┼──┼─────────────────┼──────────────────────────┤
   └─StatType──┬─Exception─┬───┘  └─StatInterval n──┘
               └─Normal────┘
```

TraceData:

```
   ┌─TraceData HEADER────────────┐  ┌─TraceRecordSize 100─┐
├──┼─────────────────────────────┼──┼─────────────────────┼─────────────────────┤
   └─TraceData──┬─FULL───────┬───┘  └─TraceRecordSize n───┘
                ├─HEADER─────┤
                ├─NONE───────┤
                └─RECORDSIZE─┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this IDSReportSet statement.

Rule: If this IDSReportSet statement is not specified inline within another statement, you must provide a *name* value. If a name is not specified for an inline IDSReportSet statement, a nonpersistent system name is created.

TypeActions
Indicates the type of actions to be taken for IDS events. The default value is no TypeActions are defined.

CONSOLE
Report IDS events to the system console.

LOG Log IDS information to the syslog daemon. Low-level detail records are optionally logged based on the LogDetail value.

STATISTICS
Log statistics to the syslog daemon based on the StatType value.

Result: Statistics are always written to the syslog INFO level.

Rule: The statistics value is applicable when the ConditionType parameter on the IDSRule statement is Attack or TR. For other ConditionType values, the STATISTICS value is ignored.

TRACE
Trace IDS information to the IDS event trace based on the TraceData value. For attack types TCP_QUEUE_SIZE, GLOBAL_TCP_STALL, and EE_XID_FLOOD, the TRACE value is ignored. No tracing is done for those attack types.

MaxEventMessage
Indicates the maximum number of event messages to be displayed on the console during a 5-minute period for an IDS attack type. Valid values are in the range 0 - 4 294 967 295. A value of 0 indicates that attack console messages are not limited. The default value is 5.

Rule: The MaxEventMessage parameter is applicable when the ConditionType parameter in the IDSRule statement is Attack. For other ConditionType values, the MaxEventMessage parameter is ignored.

LogDetail
Indicates whether detailed information is logged to the syslog daemon.

No Do not log low-level details to the syslog daemon. This is the default value.

Yes Log low-level details to the syslog daemon when detailed information is available. Low-level details are available when a scan is detected and when a Global TCP Stall attack is detected.

LoggingLevel
Indicates the syslog daemon logging level for logging IDS information. Valid values are in the range 0 - 7. The following values map to syslog daemon priority levels.

0	Emerg/Panic
1	Alert
2	Crit
3	Error
4	Warning
5	Notice
6	Info
7	Debug

The default value is 4.

StatType
Indicates the type of statistics to be gathered.

Normal
Gather all statistics. This is the default value.

Exception
Gather only exception statistics.

StatInterval
Indicates the interval length in minutes for collecting IDS statistics. Valid values are in the range 0 - 4 294 967 295. The default value is 60.

TraceData
Specifies the amount of information written to the IDS event trace.

HEADER
For IPv4 packets, trace the IP and transport headers in the packets. For IPv6 packets, trace the IPv6 header, any extension headers, and the transport header. This is the default value.

FULL Trace the entire packet.

NONE
No tracing is done.

RECORDSIZE
Trace the amount of data specified by the TraceRecordSize parameter.

TraceRecordSize
Indicates the amount in bytes of packet data to trace, when TraceData is set to RECORDSIZE. Valid values are in the range 0 - 4 294 967 295. The default value is 100.

IDSRule statement

Use the IDSRule statement to enable intrusion detection services based on the ConditionType parameter for an IDSRule statement. See the appropriate condition statement (IDSAttackCondition, IDSScanEventCondition, IDSScanGlobalCondition, or IDSTRCondition) for what is required for each IDS condition type.

Rules:

- An IDSRule statement must contain a reference to a global definition of an IDSAction statement.
- An IDSRule statement must contain one of the following references to a global or inline definition. The condition statement included must match the ConditionType parameter.
  - – IDSAttackCondition statement
  - – IDSScanEventCondition statement
  - – IDSScanGlobalCondition statement
  - – IDSTRCondition statement
- The IDS rule can contain a priority, and inline definitions or references to global definitions of the IpTimeCondition statement. An IpTimeCondition specification identifies a time period when the IDS rule is in effect.

Syntax

```
►►──IDSRule──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────┬──────────────────────────────────────────┤
   ├─┤ IDSRule Parameters ├─┤
   └─}──────────────────────┘
```

IDSRule Parameters:

```
├──┬──────────────┬──ConditionType──┬─Attack──┤ AttackCondition ├──────────┬──────►
   └─Priority n───┘                 ├─ScanEvent──┤ ScanEventCondition ├────┤
                                    ├─ScanGlobal──┤ ScanGlobalCondition ├──┤
                                    └─TR──┤ TRCondition ├──────────────────┘


                    ┌────────────────────────────┐
                    ▼                            │
►──IDSActionRef name──┬────────────────────────┬─┴──────────────────────────┤
                      ├─IpTimeCondition────────┤
                      └─IpTimeConditionRef name┘
```

AttackCondition:

```
├──┬─IDSAttackCondition──────────┬─────────────────────────────────────┤
   └─IDSAttackConditionRef name──┘
```

ScanEventCondition:

```
├──┬─IDSScanEventCondition─────────┬──────────────────────────────────────────┤
   └─IDSScanEventConditionRef name─┘
```

ScanGlobalCondition:

```
├──┬───────────────────────────────┬──────────────────────────────────────────┤
   ├─IDSScanGlobalCondition────────┤
   └─IDSScanGlobalConditionRef name┘
```

TRCondition:

```
├──┬─IDSTRCondition─────────┬─────────────────────────────────────────────────┤
   └─IDSTRConditionRef name─┘
```

Parameters

name
: A string 1 - 32 characters in length that specifies the name of this IDSRule statement.

Priority
: An integer value in the range 1 - 2 000 000 000 representing the priority associated with the rule. Only one rule is ever mapped for a given condition type. Rules are searched for a match starting at the highest priority, so if multiple rules can be matched for a given condition type, the rule with the highest priority gets matched first. If multiple rules of the same priority match, the rule that is mapped is difficult to predict.

  If this parameter is specified, the computed priority of the rule is the specified value plus 100. If this parameter is not specified, the computed priority of the rule is determined by the number of selection criteria specified, but is always less than 100.

  The selection criteria that affect the priority calculation are the following IDSAttackCondition parameters:

  - AttackType
  - RestrictedIpOptionRange or RestrictedIPOptionRangeRef
  - LocalPortRange or LocalPortRangeRef
  - RemotePortRange or RemotePortRangeRef
  - ProtocolRange or ProtocalRangeRef
  - IPv6NextHdrRange or IPv6NextHdrRangeRef
  - RestrictedIpv6OptionRange or RestrictedIpv6OptionRangeRef

  The selection criteria that affect the priority calculation are the following IDSScanEventCondition parameters:

  - LocalPortRange or LocalPortRangeRef
  - LocalHostAddr or LocalHostAddrRef
  - Protocol

  The selection criteria that affect the priority calculation are the following IDSTRCondition parameters:

  - LocalPortRange or LocalPortRangeRef

- LocalHostAddr or LocalHostAddrRef
- Protocol

Guideline: When setting the priority for multiple rules, you should not set the priority as a sequential value, for example 2, 3, 4, and 5. Instead, set the priority so there is room to change the priority, such that the rule would be preferred over another rule, without duplicating a priority. For example, you could configure the priorities as 20, 30, 40, and 50.

ConditionType

Attack Indicates that this is an Attack IDS rule. This rule is for attack detection, reporting, and prevention.

ScanEvent
Indicates that this is a scan event IDS rule. This rule defines the type of inbound traffic to be monitored by scan detection. A scan global IDS rule is also required for scan detection activation.

ScanGlobal
Indicates that this is a scan global IDS rule. This rule is for global definitions used for scan detection such as fast scan and slow scan conditions. It also defines the type of reporting used when a scan is detected.

TR Indicates that this is a traffic regulation IDS rule. This rule is for traffic regulation for TCP connections and UDP receive queues.

Result: An IDSRule should include one ConditionType parameter. If more than one is present the last one is used.

IDSAttackCondition
An inline specification of an IDSAttackCondition statement.

IDSAttackConditionRef
The name of a globally defined IDSAttackCondition statement.

IDSScanEventCondition
An inline specification of an IDSScanEventCondition statement.

IDSScanEventConditionRef
The name of a globally defined IDSScanEventCondition statement.

IDSScanGlobalCondition
An inline specification of an IDSScanGlobalCondition statement.

Rule: Only one scan global IDS rule can be configured for a stack.

IDSScanGlobalConditionRef
The name of a globally defined IDSScanGlobalCondition statement.

Rule: Only one scan global IDS rule can be configured for a stack.

IDSTRCondition
An inline specification of an IDSTRCondition statement.

IDSTRConditionRef
The name of a globally defined IDSTRCondition statement.

IDSActionRef
The name of a globally defined IDSAction statement.

IpTimeCondition
An inline specification of an IpTimeCondition statement. There is a limit of 25 IpTimeCondition specifications on the IDSRule.

IpTimeConditionRef
The name of a globally defined IpTimeCondition statement. There is a limit of 25 IpTimeCondition references on the IDSRule.

IDSScanEventCondition statement

Use the IDSScanEventCondition statement to define the conditions that scan processing monitors. These policies are searched by this ScanEvent condition type and a protocol condition of ICMP, ICMPv6, TCP, or UDP. For protocols TCP and UDP, the policy search also includes local destination port and bound IP address.

The IDSScanEventCondition statements can contain a definition of LocalHostAddr, or this value can be specified with references to global definitions of IpAddr, IpAddrSet, or IpAddrGroup statements.

The IDSScanEventCondition statements can contain an inline definition of IDSScanExclusion, or this value can be specified with references to global definitions of IDSScanExclusion statements.

The IDSScanEventCondition statements can contain a definition of LocalPortRange or this value can be specified with references to global definitions of PortRange or PortGroup statements.

Syntax

```
►►──IDSScanEventCondition──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                           └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IDSScanEventCondition Parameters ├─┤
   └─}──────────────────────────────────┘
```

IDSScanEventCondition Parameters:

```
                                          ┌─Sensitivity MEDIUM──────┐
├──┬──────────────────────────────┬──┼─────────────────────────┼────────────────────►
   ├─IDSScanExclusion─────────────┤  └─Sensitivity──┬─HIGH───┬─┘
   └─IDSScanExclusionRef name─────┘                ├─LOW────┤
                                                   ├─MEDIUM─┤
                                                   └─NONE───┘

►─Protocol──┬─┬─Tcp─┬─┤ TcpUdpScanEventData ├─┬──────────────────────────────┤
            │ └─6───┘                         │
            ├─┬─Udp─┬─┤ TcpUdpScanEventData ├─┤
            │ └─17──┘                         │
            ├─┬─Icmp─┬────────────────────────┤
            │ └─1────┘                        │
            └─┬─Icmpv6─┬──────────────────────┘
              └─58─────┘
```

TcpUdpScanEventData:

```
   ┌─LocalHostAddr All──────────────────────────┐   ┌─LocalPortRange 0────────────┐
├──┼────────────────────────────────────────────┼───┼─────────────────────────────┼──────┤
   ├─LocalHostAddr───┬─ipaddress───────────────┬┤   ├─LocalPortRange───┬─n───┬────┤
   │                 ├─ipaddress/prefixLength──┤│   │                  └─n m─┘    │
   │                 ├─ipaddress-ipaddress─────┤│   ├─LocalPortRangeRef name──────┤
   │                 └─All─────────────────────┘│   └─LocalPortGroupRef name──────┘
   ├─LocalHostAddrRef name──────────────────────┤
   ├─LocalHostAddrSetRef name───────────────────┤
   └─LocalHostAddrGroupRef name─────────────────┘
```

Parameters

name
A string 1 -32 characters in length specifying the name of this IDSScanEventCondition statement.

Rule: If this IDSScanEventCondition statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IDSScanEventCondition statement, a nonpersistent system name is created.

IDSScanExclusion
An inline specification of an IDSScanExclusion statement.

IDSScanExclusionRef
The name of a globally defined IDSScanExclusion statement.

Sensitivity
Indicates the sensitivity of events monitored for fast and slow scan detection. Events that are monitored can be classified as normal, possibly suspicious, or very suspicious. This parameter indicates which of these types of events should be counted for scan detection.

None Indicates that no events are counted.

High Indicates that all event types are counted.

Medium
Indicates that possibly suspicious and very suspicious events are counted. This is the default value.

Low Indicates that only very suspicious events are counted.

Protocol
Indicates the protocol name or number for this IDS ScanEvent rule.

Restriction: The values Icmpv6 and 58 are valid only for V1R13 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

LocalHostAddr
A local host IP address for this IDS ScanEvent rule. The specified IP address is used to match applications that either explicitly bind to this address, or that have the IP address assigned by the TCP/IP stack. All indicates than any local IP address matches this rule. The default value is All.

ipaddress
A single IPv4 or IPv6 address.

ipaddress/prefixLength
The number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and in

the range 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the defined unmasked bits.

ipaddress-ipaddress
A range of IPv4 or IPv6 addresses.

Tip: All includes all IPv4 and IPv6 addresses. If you want to include only IPv4 addresses specify LocalHostAddr 0.0.0.0/0. If you want to include only IPv6 addresses specify LocalHostAddr ::0/0.

Result: When centralized policy is used, All includes all IPv4 and IPv6 addresses regardless of the release level of the policy server.

Rule: An IPv6 address specified for LocalHostAddr cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96.

LocalHostAddrRef
The name of a globally defined IpAddr statement to be used for the local IP address specification.

LocalHostAddrSetRef
The name of a globally defined IpAddrSet statement to be used for the local IP address prefix or range specification.

LocalHostAddrGroupRef
The name of a globally defined IpAddrGroup statement to be used for the local IP address specification.

LocalPortRange
A local port for this IDS scan event rule. Valid values for *n* are 0 - 65535. If 0 is specified for *n* then the rule applies to any local port. If *n* is specified as the beginning value for a range, then 0 is not a valid value. If an *m* value is specified, then it must be greater than or equal to *n* and less than 65536. The default value is 0.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

LocalPortRangeRef
The name of a globally defined PortRange statement to be used for the local port specification

LocalPortGroupRef
The name of a globally defined PortGroup statement to be used for the local port specification.

IDSScanExclusion statement

Use the IDSScanExclusion statement to specify IP addresses and optionally, ports, that are to be excluded when monitoring for scans. For example, responses from name servers might appear to be scans, unless the name servers are excluded using this statement.

Syntax

```
►►──IDSScanExclusion──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                      └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬────────────────────────────────────────────────┤
   ├─┤ IDSScanExclusion Parameters ├─┤
   └─}──────────────────────────────┘
```

IDSScanExclusion Parameters:

```
    ┌──────────────────────────────────────────────────┐
    ▼                                                  │
├───ExcludedAddrPort──┬─ipaddress───────────────────┬──┴────────────────────────────┤
                      ├─ipaddress n─────────────────┤
                      ├─ipaddress n m───────────────┤
                      ├─ipaddress/prefixLength──────┤
                      ├─ipaddress/prefixLength n────┤
                      └─ipaddress/prefixLength n m──┘
```

Parameters

name
: A string 1-32 characters in length specifying the name of this IDSScanExclusion statement.

  Rule: If this IDSScanExclusion statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IDSScanExclusion statement, a nonpersistent system name is created.

ExcludedAddrPort
: Indicates the IP addresses and optionally ports that are to be excluded when monitoring for scans.

  ipaddress
  : A single IPv4 or IPv6 address.

  ipaddress n
  : A single IPv4 or IPv6 address and port. Valid port values for *n* are 0 - 65535. If 0 is specified for *n*, then all ports are excluded.

  ipaddress n m
  : A single IPv4 or IPv6 address and range of ports. Valid port values for *n* are 1 - 65535. The *m* value must be greater than or equal to *n* and less than 65536.

  ipaddress/prefixLength
  : An IPv4 or IPv6 prefix address specification. Valid values for the *prefixLength* value are in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the defined unmasked bits.

  ipaddress/prefixLength n
  : An IPv4 or IPv6 prefix address specification and a port. Valid values for the *prefixLength* value are in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the defined unmasked bits. Valid port values for *n* are 0 - 65535. If 0 is specified for *n*, all ports are excluded.

  ipaddress/prefixLength n m
  : An IPv4 or IPv6 prefix address specification and a range of ports. Valid values for the *prefixLength* value are in the range 0 - 32 for IPv4

addresses and 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the defined unmasked bits. Valid port values for *n* are 1 - 65535. The *m* value must be greater than or equal to *n* and less than 65536.

Result: Only the first 10000 ExcludedAddrPort entries are saved and used.

Rule: You can specify a mix of IPv4 and IPv6 addresses within one IDSScanExclusion statement.

IDSScanGlobalCondition statement

Use the IDSScanGlobalCondition statement for global scan detection and reporting. The action defines the reporting and tracing actions to take when a scan event is detected.

Rule: You can configure only one IDSRule statement with an IDSScanGlobalCondition parameter. If you configure multiple scan global rules with different names, the first instance is used and all others are discarded as errors. If you configure multiple scan global rules with the same name, the last instance is used.

Syntax

```
►►──IDSScanGlobalCondition──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                            └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────────┬───────────────────────────────────────┤
   ├─┤ IDSScanGlobalCondition Parameters ├─┤
   └─}─────────────────────────────────┘
```

IDSScanGlobalCondition Parameters:

```
   ┌─FSInterval 1─┐  ┌─FSThreshold 5─┐  ┌─SSInterval 120─┐  ┌─SSThreshold 10─┐
├──┼──────────────┼──┼───────────────┼──┼────────────────┼──┼────────────────┼──┤
   └─FSInterval n─┘  └─FSThreshold n─┘  └─SSInterval n───┘  └─SSThreshold n──┘
```

Parameters

name
: A string 1 -32 characters in length specifying the name of this IDSScanGlobalCondition statement.

  Rule: If this IDSScanGlobalCondition statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IDSScanGlobalCondition statement, a nonpersistent system name is created.

`FSInterval`
: Indicates the interval in minutes for monitoring for fast scans. Valid values are in the range 1 - 1440. The default value is 1.

`FSThreshold`
: Indicates the fast scanning threshold. A fast scan is detected if the number of

events from a single source meets or exceeds this threshold and occur within the interval defined by the FSInterval value. Valid values are in the range 1 - 64. The default value is 5.

SSInterval
Indicates the interval in minutes for monitoring for slow scans. Valid values are in the range 0 - 1440. The default value is 120. The value specified must be greater than the value specified for the FSInterval parameter. However, a value of 0 can be specified to indicate that no slow scan processing should be performed.

SSThreshold
Indicates the slow scanning threshold. A slow scan is detected if the number of events from a single source meets or exceeds this threshold and occurs within the interval defined by the SSInterval value. Valid values are in the range 0 - 64. The default value is 10. The value specified must be greater than the value specified for the FSThreshold parameter. However, a value of 0 can be specified to indicate that no slow scan processing should be performed.

IDSTRCondition statement

Use the IDSTRCondition statement for traffic regulation for TCP connections and UDP receive queues. TCP rules are mapped when a local application does a listen on a socket or when an inbound connection handshake completes. UDP rules are mapped when an inbound packet arrives at a local bound socket. UDP TR policy supersedes the TCPIP PROFILE setting of UDPQUEUELIMIT for covered ports.

Syntax

```
►►──IDSTRCondition──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                    └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬──────────────────────────────────────────┤
   └─┤ IDSTRCondition Parameters ├──┘
    └─}─────────────────────────────┘
```

IDSTRCondition Parameters:

```
   ┌─LocalHostAddr All──────────────────────┐   ┌─LocalPortRange 0───────────┐
├──┼────────────────────────────────────────┼───┼────────────────────────────┼──────►
   ├─LocalHostAddr──┬─ipaddress──────────────┬─┤   ├─LocalPortRange──┬─n───┬────┤
   │                ├─ipaddress/prefixLength─┤ │   │                 └─n m─┘    │
   │                ├─ipaddress-ipaddress────┤ │   ├─LocalPortRangeRef name─────┤
   │                └─All────────────────────┘ │   └─LocalPortGroupRef name─────┘
   ├─LocalHostAddrRef name──────────────────────┤
   ├─LocalHostAddrSetRef name───────────────────┤
   └─LocalHostAddrGroupRef name─────────────────┘
```

```
►─Protocol──┬─┬─Tcp─┬─┤ TRTcpData ├─┬──────────────────────────────────────┤
            │ └─6───┘               │
            └─┬─Udp─┬─┤ TRUdpData ├─┘
              └─17──┘
```

TRTcpData:

```
   ┌─TRtcpTotalConnections 65535─┐ ┌─TRtcpPercentage 100─┐
├──┴─TRtcpTotalConnections n─────┴─┴─TRtcpPercentage n───┴──────────►

   ┌─TRtcpLimitScope PORT_INSTANCE──────┐
►──┴─TRtcpLimitScope─┬─PORT_INSTANCE─┬─┴────────────────────────────┤
                     └─PORT──────────┘
```

TRUdpData:

```
   ┌─TRudpQueueSize VERY_LONG─────────┐
├──┴─TRudpQueueSize──┬─VERY_LONG──┬──┴─────────────────────────────┤
                     ├─LONG───────┤
                     ├─SHORT──────┤
                     └─VERY_SHORT─┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IDSTRCondition statement.

Rule: If this IDSTRCondition statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IDSTRCondition statement, a nonpersistent system name is created.

LocalHostAddr
: A local IP host address for this IDS TR rule. The specified IP address is used to match applications that either explicitly bind to this address, or that have the IP address assigned by the TCP/IP stack. All indicates that any local IP address matches this rule. The default value is All.

ipaddress
: A single IPv4 or IPv6 address.

ipaddress/prefixLength
: An IPv4 or IPv6 prefix address specification. The number of unmasked leading bits in *ipaddress*. The *prefixLength* can be in the range 0 - 32 for IPv4 addresses or 0 - 128 for IPv6 addresses. An IP address matches this condition if its unmasked bits are identical to the defined unmasked bits.

ipaddress- ipaddress
: A range of IPv4 or IPv6 addresses.

Tip: The value ALL includes all IPv4 and IPv6 addresses. If you want to include only IPv4 addresses specify LocalHostAddr 0.0.0.0/0. If you want to include only IPv6 addresses specify LocalHostAddr ::0/0.

Result: When centralized policy is used, All includes all IPv4 and IPv6 addresses regardless of the release level of the policy server.

Rules:

- The LocalHostAddr parameter cannot be specified if TRtcpLimitScope PORT was specified.
- An IPv6 address specified for LocalHostAddr cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96.

LocalHostAddrRef
The name of a globally defined IpAddr statement to be used for the local IP address specification.

LocalHostAddrSetRef
The name of a globally defined IpAddrSet statement to be used for the local IP address prefix or range specification.

LocalHostAddrGroupRef
The name of a globally defined IpAddrGroup statement to be used for the local IP address specification.

LocalPortRange
Indicates the local ports for this IDS TR rule. Valid values for *n* are 0 - 65535. If 0 is specified for *n*, the rule applies to any local port. If *n* is specified as the beginning value for a range, 0 is not a valid value. If an *m* value is specified, it must be greater than or equal to *n* and less than 65536. The default value is 0.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

LocalPortRangeRef
The name of a globally defined PortRange statement to be used for the local port specification.

LocalPortGroupRef
The name of a globally defined PortGroup statement to be used for the local port specification.

Protocol
Indicates the protocol name or number for this IDS TR rule.

TRtcpTotalConnections
Indicates the size of the total connection pool for IDS TCP traffic regulation functions. Valid values are in the range 0 - 65535. The default value is 65535.

TRtcpPercentage
Indicates the percentage of connections that can be used by a single host. The percentage is applied to the number of available connections in the pool established by TRtcpTotalConnections. Valid values are in the range 0 - 100. The default value is 100.

TRtcpLimitScope
Indicates the scope of TCP traffic regulation.

PORT_INSTANCE
Indicates that traffic regulation parameters apply to each socket that is bound to the target port individually. This is the default value.

PORT
Indicates that traffic regulation parameters apply to the aggregate of all sockets that are bound to the target port.

Rule: The LocalHostAddr parameter cannot be specified if TRtcpLimitScope PORT was specified.

TRudpQueueSize

Indicates the size of the port backlog queue. This parameter is used to select one of a number of abstract queue sizes that map to internally defined limits. For details about queue sizes, see the TR UDP information in z/OS Communications Server: IP Configuration Guide.

- LONG
- SHORT
- VERY_LONG (this is the default)
- VERY_SHORT

IPSec policy statements

This topic contains information about the following IPSec policy statements:

- "IpDataOffer statement" on page 1072
- "IpDynVpnAction statement" on page 1077
- "IpFilterGroup statement" on page 1083
- "IpFilterPolicy statement" on page 1084
- "IpFilterRule statement" on page 1087
- "IpGenericFilterAction statement" on page 1091
- "IpLocalStartAction statement" on page 1093
- "IpManVpnAction statement" on page 1098
- "IpService statement" on page 1106
- "IpServiceGroup statement" on page 1111
- "KeyExchangeAction statement" on page 1111
- "KeyExchangeGroup statement" on page 1119
- "KeyExchangeOffer statement" on page 1120
- "KeyExchangePolicy statement" on page 1126
- "KeyExchangeRule statement" on page 1131
- "LocalDynVpnGroup statement" on page 1133
- "LocalDynVpnPolicy statement" on page 1134
- "LocalDynVpnRule statement" on page 1135
- "LocalSecurityEndpoint statement" on page 1138
- "RemoteIdentity statement" on page 1144
- "RemoteSecurityEndpoint statement" on page 1146

The IPSec sample files are: /usr/lpp/tcpip/samples/pagent_CommonIPSec.conf /usr/lpp/tcpip/samples/pagent_IPSec.conf

Tip: The terms phase 1 and phase 2 refer to different types of security associations (SAs) that the z/OS IKE daemon can negotiate with its peers. Although the specific terminology for these types of security associations differs between the IKE version 1 and IKE version 2 protocols, the terms phase 1 and phase 2 refers to both versions, as shown in Table 73.

Table 73. IKE terminology: phase 1 and phase 2

Term	Usage in regard to IKE protocol version
Phase 1 security association (SA)	Refers to IKE version 1 phase 1 SAs as well as IKE version 2 IKE SAs. When a specific version is intended, that version is identified in this document.

Table 73. IKE terminology: phase 1 and phase 2 (continued)

Term	Usage in regard to IKE protocol version
Phase 2 security association (SA)	Refers to IKE version 1 phase 2 SAs as well as IKE version 2 child SAs. When a specific version is intended, that version is identified in this document.

IpDataOffer statement

Use the IpDataOffer statement to define a data offer for a dynamic VPN. An IP data offer indicates one acceptable way to protect data sent through a dynamic VPN. An IpDataOffer statement can be referenced by an IpDynVpnAction statement.

Syntax

```
►►─IpDataOffer──┬──────┬─┤ Put Braces and Parameters on Separate Lines ├──────►◄
                └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IpDataOffer Parameters ├─┤
   └─}──────────────────────────┘
```

IpDataOffer Parameters:

```
   ┌─HowToEncap──Tunnel─────────┐
├──┼────────────────────────────┼─────────────────────────────────────────────►
   └─HowToEncap──┬─Tunnel────┬──┘
                 └─Transport─┘

  ┌─HowToEncrypt──DES─────────────────────────┐
►─┼───────────────────────────────────────────┼───────────────────────────────►
  └─HowToEncrypt──┬─DES──────────────────────┬┘
                  ├─3DES─────────────────────┤
                  ├─AES──────────────────────┤
                  ├─AES_CBC KeyLength keylen─┤
                  ├─AES_GCM_16 KeyLength keylen┤
                  └─DoNot────────────────────┘

  ┌─HowToAuth ESP HMAC_MD5──────────────────┐
►─┼─────────────────────────────────────────┼─────────────────────────────────►
  └─HowToAuth──┬─AH──┬──┬─Null─────────────┬┘
               └─ESP─┘  ├─AES_GMAC_128─────┤
                        ├─AES_GMAC_256─────┤
                        ├─AES128_XCBC_96───┤
                        ├─HMAC_MD5─────────┤
                        ├─HMAC_SHA─────────┤
                        ├─HMAC_SHA1────────┤
                        ├─HMAC_SHA2_256_128┤
                        ├─HMAC_SHA2_384_192┤
                        └─HMAC_SHA2_512_256┘
  ┌─RefreshLifetimeProposed 240──────────┐
►──┤                                      ├──────────────────────────────────►
  └─RefreshLifetimeProposed proposedtime─┘

  ┌─RefreshLifetimeAccepted 120 480──────────┐
►──┤                                          ├──────────────────────────────►
  └─RefreshLifetimeAccepted mintime maxtime──┘

  ┌─RefreshLifesizeProposed None──────────────┐
►──┤                                           ├─────────────────────────────►
  └─RefreshLifesizeProposed─┬─proposedsize─┬──┘
                            └─None─────────┘

  ┌─RefreshLifesizeAccepted None─────────────────┐
►──┤                                              ├──────────────────────────┤
  └─RefreshLifesizeAccepted─┬─minsize maxsize─┬──┘
                            └─None────────────┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this IpDataOffer statement.

Rule: If this IpDataOffer statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IpDataOffer statement, a nonpersistent system name is created.

HowToEncap
The encapsulation mode of the dynamic VPN's security associations when IKE version 1 is used. The default is tunnel mode.

Tunnel
Specifies that the security association operates in tunnel mode, which protects the entire IP packet. This mode must be used for a secure tunnel between two security gateways or between a security gateway and a remote system. One or both of the communication endpoints can be on different systems than the security endpoints.

Transport
Specifies that the security association operates in transport mode, which protects only the transport-layer headers and data (for example, TCP or UDP packet) inside an IP packet. This mode can be used only when the endpoints of the security association are the two communicating systems (that is, neither system acts as a gateway).

Restriction: HowToEncap is ignored for dynamic VPNs that are negotiated using IKE version 2. The encapsulation mode for IKE version 2 security associations is determined by the HowToEncapIKEv2 parameter on the IPDynVpnAction statement.

HowToEncrypt
Encryption is done using the ESP protocol. Specify the encryption algorithm used to provide data confidentiality. The default is DES.

DES DES encryption is used with a 56–bit key and a 64–bit initialization vector.

Restriction: DES is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

3DES Triple DES runs the DES encryption algorithm three times and uses 192-bits, including 24 parity bits.

Rule: If 3DES is specified but is not supported by the system, the Policy Agent fails the policy.

AES Deprecated and treated as a synonym for AES_CBC KeyLength 128.

Rule: If AES is specified but AES encryption in CBC mode is not supported by this TCP/IP stack, Policy Agent fails the policy.

AES_CBC
The Advanced Encryption Standard (AES) algorithm is used in Cipher Block Chaining (CBC) mode.

Rules:

- The key length is measured in bits, and a *keylen* of either 128 or 256 must be specified.
- If AES_CBC is specified but AES encryption in CBC mode is not supported by this TCP/IP stack, Policy Agent fails the policy.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

AES_GCM_16
The AES algorithm is used in Galois Counter Mode (GCM) and with a 16-byte Integrity Check Value (ICV). Galois Counter Mode is a combined-mode algorithm that performs both encryption and authentication simultaneously.

Rules:

- The key length is measured in bits, and a *keylen* of either 128 or 256 must be specified.
- HowToAuth ESP NULL must be specified if AES_GCM_16 is specified.
- If AES_GCM_16 is specified but AES encryption in Galois Counter Mode is not supported by this TCP/IP stack, Policy Agent fails the policy.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

DoNot
No encryption is used.

If the HowToEncrypt value **DoNot** is specified with a HowToAuth **ESP** value, the ESP header is present, but the payload is not encrypted (ESP_NULL).

HowToAuth
The desired authentication policy indicating which protocol and which algorithm to use when authenticating data. The default is ESP HMAC_MD5.

AH Carry authentication in AH headers.

ESP Carry authentication in ESP headers.

AES_GMAC_128
Use the AES_GMAC algorithm to encode authentication data in either AH or ESP headers, with 128-bit keys. AES_GMAC functions as a combined-mode algorithm that provides authentication without encryption.

Rule: AES_GMAC_128 may be specified only in combination with HowToEncrypt DoNot.

Tip: If you want a combined-mode algorithm that provides both authentication and encryption, choose HowToEncrypt AES_GCM_16.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

AES_GMAC_256
Use the AES_GMAC algorithm to encode authentication data in either AH or ESP headers, with 256-bit keys. AES_GMAC functions as a combined-mode algorithm that provides authentication without encryption.

Rule: AES_GMAC_256 may be specified only in combination with HowToEncrypt DoNot.

Tip: If you want a combined-mode algorithm that provides both authentication and encryption, choose HowToEncrypt AES_GCM_16.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

AES128_XCBC_96
Use the AES128_XCBC algorithm to encode authentication data in either AH or ESP headers, with 128-bit keys and hash truncation to 96 bits.

Restriction: AES128_XCBC_96 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HMAC_MD5
Use the HMAC_MD5 algorithm to encode authentication data in either AH or ESP headers.

Restriction: HMAC_MD5 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

HMAC_SHA
Deprecated and treated as a synonym for HMAC_SHA1.

HMAC_SHA1
Use the HMAC_SHA1 algorithm to encode authentication data in either AH or ESP headers.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HMAC_SHA2_256_128
Use the HMAC_SHA2_256 algorithm to encode authentication data in either AH or ESP headers, with 256-bit keys and hash truncation to 128 bits.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HMAC_SHA2_384_192
Use the HMAC_SHA2_384 algorithm to encode authentication data in either AH or ESP headers, with 384-bit keys and hash truncation to 192 bits.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HMAC_SHA2_512_256
Use the HMAC_SHA2_512 algorithm to encode authentication data in either AH or ESP headers, with 512-bit keys and hash truncation to 256 bits.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Null Use no authentication.

Rule: NULL may only be specified with HowToAuth ESP and only in combination with HowToEncrypt AES_GCM_16.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Restriction: HowToAuth AH may be specified in combination with HowToEncrypt only if the security association is negotiated using IKE version 1 because IKE version 2 does not permit the combining of AH and ESP. If this combination is specified for an IKE version 2 security association, the IPDataOffer is ignored.

RefreshLifetimeProposed
The security association lifetime in minutes. For IKE version 1, this value is proposed when acting as the initiator of a key exchange negotiation. For IKE version 2, this value determines the refresh lifetime.The default is 240.

proposedtime
The lifetime proposed (for IKE version 1) or used (for IKE version 2) for the phase 2 tunnel. Valid values are in the range 1 - 9 999. The proposed lifetime value should be within the range specified by the RefreshLifetimeAccepted parameter.

Tip: For IKE version 2 security associations, if the RefreshLifetimeProposed value is longer than that of the ReauthInterval on the associated KeyExchangeAction, the reauthentication will usually occur before the re-keying occurs.

RefreshLifetimeAccepted
A range of acceptable security association lifetimes in minutes. This range is accepted when acting as the responder of IKE version 1 key exchange negotiation. The default is 120 480.

mintime
The minimum lifetime that can be accepted.

maxtime
The maximum lifetime that can be accepted. This value must be ≥ to the *mintime* value.

Valid values for each option are in the range 1 - 9 999.

Restriction: This parameter is ignored for IKE version 2 SAs.

RefreshLifesizeProposed
The security association lifesize in Kbytes. If a *proposedsize* value is specified, then this value is proposed when acting as the IKE version 1 initiator of a key exchange negotiation. For IKE version 2, this value determines the refresh lifesize. If **None** is specified, then no lifesize is proposed for IKE version 1 or used for IKE version 2. The default is **None**.

proposedsize
The proposed lifesize for the negotiation. Valid values are in the range 1 - 4 194 300. The proposed *lifesize* value should be within the range specified by RefreshLifesizeAccepted parameter, if that parameter is not specified as **None**.

None No lifesize should be proposed for IKE version 1 or used for IKE version 2. If this parameter is specified as **None**, then RefreshLifesizeAccepted parameter should also be specified as **None**.

RefreshLifesizeAccepted
The security association lifesize in Kbytes. If *minsize* and *maxsize* values are specified, then this range is accepted when acting as the responder of an IKE version 1 key exchange negotiation. If **None** is specified, then no *lifesize* is accepted when acting as the responder of a key exchange negotiation. The default is **None**.

Result: The IKED accepts a proposed *lifesize* greater than 4 194 300, only if a *maxsize* of exactly 4 194 300 is specified. If the IKED accepts a *lifesize* greater than 4 194 300, it assigns an actual lifesize of 4 194 300 to the security association.

minsize
The minimum *lifesize* that can be accepted.

maxsize
The maximum *lifesize* that can be accepted. This value must be ≥ to the *minsize* value.

None No *lifesize* is accepted. If this parameter is specified as **None**, then RefreshLifesizeProposed value should also be specified as **None**.

Valid values for the *minsize* and *maxsize* options are 1 - 4 194 300.

Restriction: This parameter is ignored for IKE version 2 SAs.

Rules:
- The IpDataOffer statement allows for 0 - 1 authentication proposals and 0 - 1 encryption proposals. Multiple authentication proposals or multiple encryption proposals cannot be specified in the same IpDataOffer statement.
- To propose authentication using only the ESP header, specify **HowToEncrypt DoNot** and **HowToAuth ESP xxx** where xxx represents a valid authentication algorithm.. The ESP header is present, but the payload is not encrypted.

Result: If both HowToAuth AH and HowToEncrypt are specified when using IKE version 1, then for outbound traffic the encryption proposal is always applied before the authentication proposal (IKE version 2 does not permit combining AH and ESP). For example, if **HowToEncrypt DES** and **HowToAuth AH HMAC_SHA** are specified, this is understood to mean that data is first encrypted and the results are then authenticated and carried in an AH header.

IpDynVpnAction statement

Use the IpDynVpnAction statement to indicate how selected data traffic between two security endpoints should be protected utilizing dynamically established security associations. An IpDynVpnAction statement contains inline definitions or references to IpDataOffer statements, or both.

Dynamically established security associations are created by the IKE daemon and are used to protect data sent through a dynamic VPN. Dynamic VPNs can be established in the following ways:

- Automatically established when the TCP/IP stack comes up or when the IKE daemon comes up, or both. This is known as an autoactivation.
- Established with an **ipsec** command. This is known as a command-line activation.
- Automatically established with an outbound IP packet. This is known as an on-demand activation.
- Established with a phase 2 IKE negotiation initiated by a remote security endpoint. This is known as a remote activation.

Syntax

```
►►──IpDynVpnAction──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IpDynVpnAction Parameters ├─┤
   └─}─────────────────────────────┘
```

IpDynVpnAction Parameters:

```
   ┌─Pfs None──────────┐   ┌─Initiation Either───────────┐   ┌─VpnLife 1440─┐
├──┴─Pfs──┬─Group1──┬──┴───┴─Initiation──┬─LocalOnly──┬─┴───┴─VpnLife n────┴──────►
          ├─Group2──┤                    ├─RemoteOnly─┤
          ├─Group5──┤                    └─Either─────┘
          ├─Group14─┤
          └─None────┘

                                         ┌──────────────────────────────────┐
   ┌─InitiateWithPfs None─────────────┐  ▼ ┌─AcceptablePfs None───────────┐ │
►──┴─InitiateWithPfs──┬─Group1──┬─────┴────┴─AcceptablePfs──┬─Group1──┬──┴─┴──────►
                      ├─Group2──┤                           ├─Group2──┤
                      ├─Group5──┤                           ├─Group5──┤
                      ├─Group14─┤                           ├─Group14─┤
                      ├─Group19─┤                           ├─Group19─┤
                      ├─Group20─┤                           ├─Group20─┤
                      ├─Group21─┤                           ├─Group21─┤
                      ├─Group24─┤                           ├─Group24─┤
                      └─None────┘                           └─None────┘

   ┌─HowToEncapIKEv2 Either─────────────┐  ┌─PassthroughDF Yes────────────────────┐
►──┴─HowToEncapIKEv2──┬─Tunnel────┬─────┴──┴─PassthroughDF──┬─No──┬─Clear─┬─┬──┴──►
                      ├─Transport─┤                         │     ├─Set───┤ │
                      └─Either────┘                         │     └─Clear─┘ │
                                                            └─Yes───────────┘
```

```
                                           ┌─────────────────────────┐
    ┌─PassthroughDSCP Yes──────┐           ▼                         │
►───┼──────────────────────────┼───────────┬─IpDataOffer──────────┬──┴──────────────────────┤
    └─PassthroughDSCP──┬─No──┬─┘           └─IpDataOfferRef name──┘
                       └─Yes─┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IpDynVpnAction statement. The name cannot start with a dash (-) or contain any commas (,).

Pfs
: Specifies whether perfect forward secrecy (PFS) is used when negotiating the security association, and if so, what Diffie-Hellman group is used. The default is None.

None Do not use perfect forward secrecy.

Group1
Modular exponentiation group with a 768-bit modulus.

Restriction: Group 1 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group2
Modular exponentiation group with a 1024-bit modulus.

Restriction: Group 2 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group5
Modular exponentiation group with a 1536-bit modulus.

Restriction: Group 5 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group14
Modular exponentiation group with a 2048-bit modulus.

Guideline: If you are using encryption or authentication algorithms with a 128-bit key, use Diffie-Hellman groups 5,14,19,20, or 24. If you are using encryption or authentication algorithms with a key length of 256 bits or greater, use Diffie-Hellman group 21.

Rule: Pfs is deprecated. Use InitiateWithPfs and AcceptablePfs parameters instead. If you use Pfs, then InitiateWithPfs and AcceptablePfs are set to the Pfs value.

Restriction: Do not use the Pfs parameter in conjunction with the InitiateWithPfs or AcceptablePfs parameters.

Initiation
: Specifies which system can initiate the security associations for this dynamic tunnel. The default is **Either**.

LocalOnly
Specifies that this system must initiate the negotiation.

RemoteOnly
Specifies that another system must initiate the negotiation.

Either Specifies that this system can either initiate a negotiation or respond to a negotiation initiated by another system.

`VpnLife`

Maximum length of time that phase 2 SAs should continue to be refreshed, in minutes. The VpnLife parameter is set for a dynamic VPN tunnel when the first SA is established for the tunnel. The VpnLife value for a dynamic VPN tunnel can be changed by deactivating the tunnel, changing the configured VpnLife value, and then activating the tunnel. Valid values are in the range 0 - 525 600 minutes. Specifying a value of 0 means that the maximum lifetime is infinite. The default is 1440 minutes. In any case, regardless of the VpnLife setting, SAs that are configured with AllowOnDemand Yes may cease to be refreshed if they have not protected any network traffic during their previous refresh interval.

`AcceptablePfs`

Specifies acceptable Diffie-Hellman groups to use for perfect forward secrecy (PFS). The default is **None**.

None Do not use perfect forward secrecy.

Group1

Modular exponentiation group with a 768-bit modulus.

Restriction: Group 1 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group2

Modular exponentiation group with a 1024-bit modulus.

Restriction: Group 2 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group5

Modular exponentiation group with a 1536-bit modulus.

Restriction: Group 5 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group14

Use modular exponentiation group with a 2048-bit modulus.

Group19

Random 256-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group20

Random 384-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group21

Random 521-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group24

Modular exponentiation group with a 2048-bit modulus and 256-bit prime order subgroup.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Result: For negotiations using IKE version 1, the AcceptablePfs list is used when the z/OS IKE daemon is the responder for a security association. For negotiations using IKE version 2, the AcceptablePfs list is used in both initiator and responder modes.

Guideline: If you are using encryption or authentication algorithms with a 128-bit key, use Diffie-Hellman groups 5,14,19,20, or 24. If you are using encryption or authentication algorithms with a key length of 256 bits or greater, use Diffie-Hellman group 21.

Rule: The InitiateWithPfs Diffie-Hellman group must be specified as one of the values in the AcceptablePfsList parameter.

Restriction: Do not use the Pfs parameter in conjunction with the InitiateWithPfs or AcceptablePfs parameters.

`InitiateWithPfs`

Specifies whether perfect forward secrecy (PFS) is used as initiator of the security association, and if so, what Diffie-Hellman group is used. The default is None.

None Do not use perfect forward secrecy.

Group1

Modular exponentiation group with a 768-bit modulus.

Restriction: Group 1 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group2

Modular exponentiation group with a 1024-bit modulus.

Restriction: Group 2 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group5

Modular exponentiation group with a 1536-bit modulus.

Restriction: Group 5 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group 14

Modular exponentiation group with a 2 048-bit modulus.

Group19

Random 256-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group20

Random 384-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group21

Random 521-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group24
Modular exponentiation group with a 2048-bit modulus and 256-bit prime order subgroup.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Result: For negotiations using IKE version 1, the InitiateWithPfs selection is used when sending the proposal. For negotiations using IKE version 2, all PFS selections specified on the AcceptablePfs list are included when sending the proposal, but the InitiateWithPfs selection is sent as the first choice.

Guideline: If you are using encryption or authentication algorithms with a 128-bit key, use Diffie-Hellman groups 5,14,19,20, or 24. If you are using encryption or authentication algorithms with a key length of 256 bits or greater, use Diffie-Hellman group 21.

HowToEncapIKEv2
Specifies the encapsulation mode to use when establishing IKE version 2 security associations. The default is Either.

Either Support both tunnel and transport mode. When initiating the security association, the choice is made based on the topology of the connection. Specifically:

- If both IKE peers are hosts, then a transport mode SA is proposed. If the peer accepts the use of transport mode, then transport mode is used for the SA. If the peer responds requiring tunnel mode, then tunnel mode is used for the SA.

  Tip: Specify the Transport keyword on the HowToEncapIKEv2 parameter if you want to reject the SA if the peer responds requiring tunnel mode.
- If either IKE peer is a gateway, then tunnel mode is used.

Use the mode proposed by the initiator when responding to a negotiation that was initiated by an IKE peer.

Transport
Supports only transport mode security associations. When initiating the SA, the behavior depends on the topology of the connection. Specifically:

- If both IKE peers are hosts, then a transport mode SA is proposed. If the peer accepts the use of transport mode, then transport mode is used for the SA. If the peer responds requiring tunnel mode, then the SA is deactivated and an error message logged.

  Tip: Specify the Either keyword on the HowToEncapIKEv2 parameter if you want to propose transport mode but are willing to use tunnel mode if the peer responds requiring tunnel mode.
- If either IKE peer is a gateway, the local activation fails and an error message is logged.

When responding to a remote initiation, if the initiator requests tunnel mode, the negotiation is rejected with a NO_PROPOSAL_CHOSEN notification.

Tunnel
Supports only tunnel mode security associations. When initiating, only tunnel mode is proposed. When responding to a remote initiation, if

the initiator requests transport mode, IKE responds without acknowledging the transport mode notification, thus forcing a tunnel mode SA to be established.

Restriction: The HowToEncapIKEv2 parameter is ignored when negotiating IKE version 1 tunnels. The encapsulation mode for IKE version 1 security associations is determined by the HowToEncap value on the selected IpDataOffer.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

PassthroughDF
When this parameter is set to No, the do not fragment bit is set to 0 (if the value Clear is specified) or 1 (if the value Set is specified) on the outer IP header for an IPv4 tunnel mode SA. When this parameter is set to Yes, the do not fragment bit is copied from the inner IP header to the outer IP header for an IPv4 tunnel mode SA. This parameter's setting is ignored for IPv6 or transport mode SAs.

Restriction: This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

PassthroughDSCP
When this parameter is set to No, the Differentiated Services Code Point (DSCP) field is set to 0 on the outer IP header for a tunnel mode SA. When this parameter is set to Yes, the DSCP field is copied from the inner IP header to the outer IP header for a tunnel mode SA. This setting is ignored for transport mode SAs.

Restriction: This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

IpDataOffer
An inline specification of an IpDataOffer statement to be used when initiating a phase 2 negotiation.

Restriction: A IpDynVpnAction statement is limited to a maximum of 48 IpDataOffer or IpDataOfferRef statements.

IpDataOfferRef
The name of a globally defined IpDataOffer statement to be used when initiating a phase 2 negotiation.

Restriction: A IpDynVpnAction statement is limited to a maximum of 48 IpDataOffer or IpDataOfferRef statements.

IpFilterGroup statement

Use the IpFilterGroup statement to define an IP filter group. An IpFilterGroup statement identifies a set of IpFilterRule statements that make up the IP filter group. An IpFilterGroup statement can be referenced by an IpFilterPolicy statement.

Syntax

```
►►──IpFilterGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬────────────────────────────────────────────────────┤
   │   ┌──────────────────────────────┐ │
   │   ▼                              │ │
   ├─────┬─IpFilterRuleRef name─┬─────┴─┤
   │     └─IpFilterRule─────────┘       │
   └─}──────────────────────────────────┘
```

Parameters

name
: A string of 1–32 characters for the name of this IP Filter Group.

IpFilterRuleRef
: The name of a globally defined IpFilterRule statement to be included in the group.

IpFilterRule
: An inline specification of an IpFilterRule to be included in this group.

IpFilterPolicy statement

Use the IpFilterPolicy statement to define an IP filter policy. The IpFilterPolicy statement can contain a combination of references to IpFilterGroup statements and IpFilterRule statements and inline IpFilterRule statements.

Communication Server's integrated IP filtering is enabled for a stack when IPSECURITY is specified on the IPCONFIG statement of that stack's TCP/IP profile. When Communication Server's integrated IP filtering is enabled, IP packets are subject to the IP filters generated by the applicable IpFilterPolicy statement. IP filters are generated in the order specified on the IpFilterPolicy statement. If a reference to an IpFilterGroup statement is encountered, all the IP filters for that group are generated in the order referenced by the IpFilterGroup statement.

Requirement: The IpFilterPolicy statement is required in order to define IP filters to the Policy Agent.

The IpFilterPolicy statement can appear in the common IPSec policy file, a stack-specific IPSec policy file, or both. If it appears in both, Policy Agent only uses the statement contained in the stack-specific IPSec policy file. It should appear at most once only in each file. If it appears multiple times in a file, the last one encountered is used.

Syntax

```
►►──IpFilterPolicy──┤ Put Braces and Parameters on Separate Lines ├──────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{────────────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IPFilterPolicy Parameters ├────┤
   └─}────────────────────────────────┘
```

IPFilterPolicy Parameters:

```
   ┌─PreDecap On────────────┐  ┌─FilterLogging Off───────────┐
├──┼────────────────────────┼──┼─────────────────────────────┼──────────────────────────►
   └─PreDecap──────┬─On──┬──┘  └─FilterLogging──────┬─On──┬──┘
                   ├─Off─┤                          ├─Off─┤
                   ├─Yes─┤                          ├─Yes─┤
                   └─No──┘                          └─No──┘

   ┌─IpFilterLogImplicit No────────┐   ┌─AllowOnDemand No──────────┐
►──┼───────────────────────────────┼───┼───────────────────────────┼──────────────────►
   └─IpFilterLogImplicit──────┬─Yes─┬┘  └─AllowOnDemand──────┬─Yes─┬┘
                              ├─No──┤                       ├─No──┤
                              ├─On──┤                       ├─On──┤
                              └─Off─┘                       └─Off─┘

   ┌─ImplicitDiscardAction Silent──────┐   ┌─FIPS140 No─────────────┐
►──┼───────────────────────────────────┼───┼────────────────────────┼──────────────►
   └─ImplicitDiscardAction──┬─ICMP───┬─┘   └─FIPS140──────┬─Yes─┬───┘
                            └─Silent─┘                   ├─No──┤
                                                         ├─On──┤
                                                         └─Off─┘

   ┌─────────────────────────────┐
   ▼                             │
►────┬─IpFilterRule───────────┬──┴────────────────────────────────────────────────────┤
     ├─IpFilterRuleRef name───┤
     └─IpFilterGroupRef name──┘
```

Parameters

PreDecap
Indicates whether AH/ESP packets should be filtered before being decapsulated.

FilterLogging
Indicates whether packet filter logging is enabled or disabled. The log messages controlled by this parameter are EZD0814I, EZD0815I, EZD0821I, EZD0832I, EZD0833I, EZD0836I, and EZD0822I.

If FilterLogging is enabled, then the log setting on the individual filter rules is honored. The log setting for individual rules is specified with the IpFilterLogging parameter on the IpGenericFilterAction statement referenced by the IpFilterRule statement.

If FilterLogging is disabled, then the log setting on the individual filter rules is ignored and no packet filter logging is done.

IpFilterLogImplicit
Indicates whether packet filter logging should be done for packets that are denied by the implicit deny all rule at the end of the filter table. If a packet does not match any of the filter rules defined in Policy Agent, then the packet is denied by an implicit deny all rule. Logging is done for this deny if the value of the IpFilterLogImplicit parameter is Yes and FilterLogging is enabled.

AllowOnDemand
Indicates whether OnDemand negotiations of security associations should be allowed for the case where an IpLocalStartAction is not referenced from the IpFilterRule.

ImplicitDiscardAction
Indicates the discard action that is to be applied to packets that are denied by the implicit deny all rule at the end of the filter table. If a packet does not match any of the filter rules defined in Policy Agent, then the packet is denied by an implicit deny all rule.

Silent Specify this value to discard the packet silently.

ICMP Specify this value to send an ICMP or ICMPv6 destination unreachable error with reason `administratively prohibited` to the origin of the discarded packet. ICMP errors are not generated for locally originated traffic; they are generated only for remote traffic that is being received or forwarded.

Guideline: If you specify ImplicitDiscardAction ICMP, you should create a filter rule permitting these ICMP errors.

Restriction: This parameter is valid only for V1R10 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

FIPS140
Specifies whether the TCP/IP stack should perform cryptographic operations by invoking cryptographic modules that are designed to meet the Level 1 security requirements documented in the Federal Information Processing Standard (FIPS) publication 140 (FIPS 140).

Yes or On
Perform all IPSec-related TCP/IP cryptographic operations using cryptographic modules that are designed to meet FIPS 140 requirements. When the value of yes or on is specified, the TCP/IP stack is running in FIPS 140 mode.

No or Off
The TCP/IP stack might perform some IPSec-related cryptographic operations using cryptographic modules that do not adhere to the FIPS 140 requirements. When the value of no or off is specified, the TCP/IP stack is not running in FIPS 140 mode.

Rule: The FIPS140 parameter may not be modified while the TCP/IP stack is running. Attempts to change the FIPS140 setting while the TCP/IP stack is running will be treated as policy configuration errors by Policy Agent.

Tip: Enabling FIPS 140 mode provides a higher degree of assurance of the integrity of the cryptographic modules that the TCP/IP stack uses, including ICSF and System SSL. However, enabling FIPS 140 mode might require additional setup and configuration, it will restrict the available set of cryptographic algorithms, and it might result in a reduction in performance. See Cryptographic standards and FIPS 140 in z/OS Communications Server: IP Configuration Guide for more information.

Restriction: This parameter is valid only for V1R12 and later releases. See “General syntax rules for Policy Agent” on page 933 for details

IpFilterRule
An inline specification of an IpFilterRule statement to be included in the policy.

IpFilterRuleRef
The name of a globally defined IpFilterRule statement to be included in the policy.

IpFilterGroupRef
The name of a globally defined IpFilterGroup statement to be included in the policy.

Result: If the IpFilterPolicy statement is deleted, then all IpFilter policies are deleted from the corresponding stack. The stack reverts to using the filter policy defined using the IPSEC statement in the TCP/IP profile. Any IpLocalStartAction actions contained in the IpFilterPolicy statement are deleted from the IKE daemon.

IpFilterRule statement

Use the IpFilterRule statement to define one or more IP filters.

The information provided on the IpFilterRule statement is combined to generate IP filters. An IpFilterRule statement that is globally defined can be referenced by an IpFilterPolicy statement and an IpFilterGroup statement.

A generated IP filter consists of a source and destination IP address specification, a service specification, an optional time period specification, a security action, and an optional local start action. The policy condition is formed by combining IP address information with port, protocol, security class, direction, and routing information from the IpService statement or the IpServiceGroup statement. An IpTimeCondition statement identifies when the generated IP filter is in effect. Security actions include the generic (permit, deny, or ipsec) action (IpGenericFilterAction), the manual VPN tunnel action (IpManVpnAction) and the dynamic VPN tunnel action (IpDynVpnAction). The optional local start action (IpLocalStartAction) is used for local on-demand or command-line activation of dynamic VPN tunnels.

Syntax

```
►►──IpFilterRule──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├───┬─{──────────────────────────┬──────────────────────────────────────────────┤
    ├─┤ IPFilterRule Parameters ├─┤
    └─}──────────────────────────┘
```

IPFilterRule Parameters:

```
   ┌─IpSourceAddr All─────────────────────────┐   ┌─IpDestAddr All─────────────────────────┐
├──┼──────────────────────────────────────────┼───┼────────────────────────────────────────┼──►
   ├─IpSourceAddr──┬─ipaddress───────────────┬┤   ├─IpDestAddr──┬─ipaddress───────────────┬┤
   │               ├─ipaddress/prefixLength──┤│   │             ├─ipaddress/prefixLength──┤│
   │               ├─ipaddress-ipaddress─────┤│   │             ├─ipaddress-ipaddress─────┤│
   │               ├─All─────────────────────┤│   │             ├─All─────────────────────┤│
   │               ├─All4────────────────────┤│   │             ├─All4────────────────────┤│
   │               └─All6────────────────────┘│   │             └─All6────────────────────┘│
   ├─IpSourceAddrRef name─────────────────────┤   ├─IpDestAddrRef name─────────────────────┤
   ├─IpSourceAddrSetRef name──────────────────┤   ├─IpDestAddrSetRef name──────────────────┤
   └─IpSourceAddrGroupRef name────────────────┘   └─IpDestAddrGroupRef name────────────────┘
```

```
                                   ┌──────────────────┐  ┌──────────────────────────┐
                                   ▼                  │  ▼                          │
►──┬────────────────────────┬──────┬──────────────────┬──┬─IpService──────────────┬──────────────►
   ├─RemoteIdentity─────────┤      └─IpTimeCondition──┘  ├─IpServiceRef name──────┤
   └─RemoteIdentityRef name─┘                            └─IpServiceGroupRef name─┘

►─IpGenericFilterActionRef name──┬─────────────────────────────────────────────────────┬──────────┤
                                 ├─IpManVpnActionRef name──────────────────────────────┤
                                 └─IpDynVpnActionRef name──┬──────────────────────────┬┘
                                                           └─IpLocalStartActionRef name┘
```

Parameters

name
A string of 1–32 characters specifying the name of this IpFilterRule statement. The name cannot start with a dash (-) or contain any commas (,).

IpSourceAddr
A source IP address specification.

ipaddress
A single IP address indicating the source address that must be contained in an IP packet for this rule's action to be performed.

ipaddress/prefixLength
A prefix address specification indicating the applicable source IP addresses that can be contained in an IP packet for this rule's action to be performed. The *prefixLength* is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP packet matches this condition if its source address unmasked bits are identical to the defined unmasked bits.

ipaddress-ipaddress
A range of IP addresses indicating applicable source addresses that can be contained in an IP packet for this rule's action to be performed.

All Indicates that any source IPv4 address can be contained in an IP packet for this rule's action to be performed. **All** and **All4** are interchangeable values.

All4 Indicates that any source IPv4 address can be contained in an IP packet for this rule's action to be performed.

All6 Indicates that any source IPv6 address can be contained in an IP packet for this rule's action to be performed.

IpSourceAddrRef
The name of a globally defined IpAddr statement to be used for the source IP address specification.

IpSourceAddrSetRef
The name of a globally defined IpAddrSet statement to be used for the source IP address prefix or range specification.

IpSourceAddrGroupRef
The name of a globally defined IpAddrGroup statement to be used for the source IP address specification.

IpDestAddr
A destination IP address specification.

ipaddress
A single IP address indicating the destination address that must be contained in an IP packet for this rule's action to be performed.

ipaddress/prefixLength
A prefix address specification indicating the applicable destination IP addresses that can be contained in an IP packet for this rule's action to be performed. The *prefixLength* value is the number of unmasked leading bits in the specified *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP packet matches this condition if its destination address unmasked bits are identical to the defined unmasked bits.

ipaddress-ipaddress
A range of IP addresses indicating applicable destination addresses that can be contained in an IP packet for this rule's action to be performed.

All Indicates that any destination IPv4 address can be contained in an IP packet for this rule's action to be performed. **All** and **All4** are interchangeable values.

All4 Indicates that any destination IPv4 address can be contained in an IP packet for this rule’s action to be performed.

All6 Indicates that any destination IPv6 address can be contained in an IP packet for this rule’s action to be performed.

IpDestAddrRef
The name of a globally defined IpAddr statement to be used for the destination IP address specification.

IpDestAddrSetRef
The name of a globally defined IpAddrSet statement to be used for the destination IP address prefix or range specification.

IpDestAddrGroupRef
The name of a globally defined IpAddrGroup statement to be used for the destination IP address specification.

IpTimeCondition
An inline specification of an IpTimeCondition statement. There is a limit of 25 IpTimeCondition specifications and references on the IpFilterRule statement.

RemoteIdentity
An inline specification of a RemoteIdentity statement. If specified, the RemoteIdentity value limits traffic that matches this filter rule. Only IPSec traffic for which the remote IKE identity matches or is contained by the RemoteIdentity matches this filter rule.

Rules:

- You can specify the RemoteIdentity parameter only for filter rules that reference an IpDynVpnAction statement.
- This parameter requires a remote activation so that the user's identity and location become known.
- Because local activations are not valid, you cannot specify the RemoteIdentity parameter for filter rules that reference an IpLocalStartAction statement.

Tip: Specify the RemoteIdentity for mobile users whose IKE identity is known but whose IP address is unknown or unpredictable.

Guideline: When you create an IpFilterRule and you specify RemoteIdentity, specify FilterByIdentity Yes on the KeyExchangeAction statement for the corresponding KeyExchangeRule statement. When you create an IPSec IpFilterRule without a RemoteIdentity, specify FilterByIdentity No on the KeyExchangeAction statement for the corresponding KeyExchangeRule statement.

Restriction: This parameter, as well as the RemoteIdentityRef parameter, is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

RemoteIdentityRef
The name of a globally defined RemoteIdentity statement.

IpTimeConditionRef
The name of a globally defined IpTimeCondition statement. There is a limit of 25 IpTimeCondition specifications and references on the IpFilterRule statement.

IpService
An inline specification of an IpService statement.

IpServiceRef
The name of a globally defined IpService statement.

IpServiceGroupRef
The name of a globally defined IpServiceGroup statement.

IpGenericFilterActionRef
The name of a globally defined IpGenericFilterAction statement.

IpManVpnActionRef
The name of a globally defined IpManVpnAction statement.

Rule: If a manual tunnel should be used to provide IPSec protection of the data, then an IpManVpnAction reference is needed in addition to the IpGenericFilterAction reference. The IpGenericFilterAction reference must specify an IpFilterAction value of IpSec.

IpDynVpnActionRef
The name of a globally defined IpDynVpnAction statement.

Rule: If a dynamic tunnel should be used to provide IPSec protection, then an IpDynVpnAction reference is needed in addition to the IpGenericFilterAction reference. The IpGenericFilterAction must specify an IpFilterAction value of IpSec.

IpLocalStartActionRef
The name of a globally defined IpLocalStartAction statement.

Requirement: An IpLocalStartAction statement can be specified only in conjunction with an IpDynVpnAction statement. The IpLocalStartAction statement is required if the dynamic VPN is not a host-to-host configuration and is locally activated.

Results:

- If the IpSourceAddrGroupRef, IpDestAddrGroupRef, or IpServiceGroupRef statement is specified, multiple filters might be generated. If more than one inline or referenced IpService statement is specified, multiple filters might be generated. If the associated IpService is bidirectional, then multiple filters are generated. In this case, the base name has a number appended to uniquely identify the generated filters.

- On an **ipsec -f display -n** IpFilterRuleName command, all IP filter rules with a base name matching the IpFilterRuleName value are displayed.

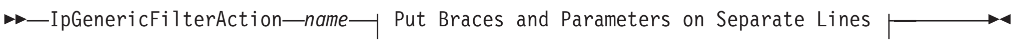
Guideline: The IP address of a remote system, represented by a filter rule's destination address specification, is always a public address when the peer is behind a NAT device. The NAT device uses the private IP address of the peer to choose a public address and replaces it in the IP header. If the peer system is behind a security gateway that is behind a NAT device, the NAT device uses the private IP address of the gateway to choose a public address because the gateway first encapsulates the peer packet in a packet with the private address of the gateway.

Rules:
- Filter rules that reference an IpManVpnAction or IpDynVpnAction statement must have a direction of bidirectional specified in the IpService statement.
- All IpFilterRule statement addresses must be in the same address family (IPv4 or IPv6).
- For any IpFilterRule statement, all of its associated actions and the associated IP addresses must be in the same address family (IPv4 or IPv6).

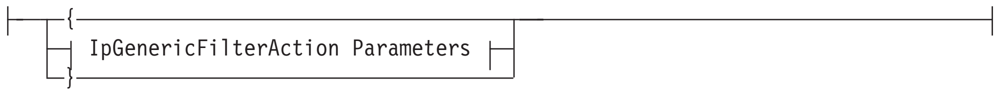
IpGenericFilterAction statement

Use the IpGenericFilterAction statement to indicate whether selected traffic should be denied, permitted, or permitted with IPSec protection. It is also used to indicate actions (for example, logging) that are applicable to both IPSec and non-IPSec traffic.

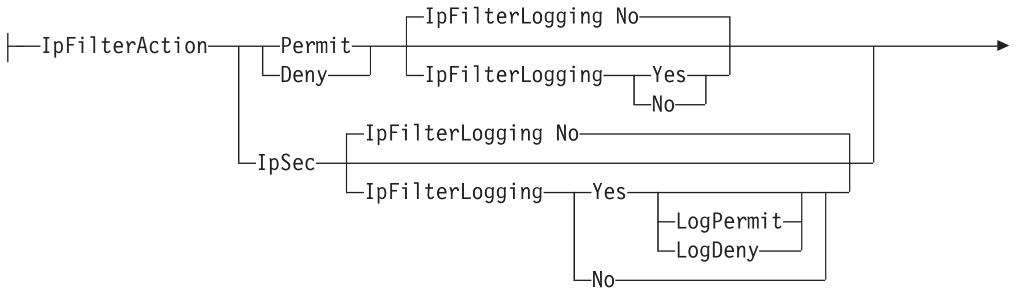
Syntax



Put Braces and Parameters on Separate Lines:



IpGenericFilterAction Parmeters:



```
   ┌─DiscardAction Silent──────┐
►──┼───────────────────────────┼──────────────────────────────────────────────┤
   └─DiscardAction──┬─ICMP───┬─┘
                    └─Silent─┘
```

Parameters

name

A string 1 - 32 characters in length specifying the name of this IpGenericFilterAction statement. The name cannot start with a dash (-) or contain any commas (,).

IpFilterAction

Indicates the action that should be applied to a packet matching this rule.

Permit
Traffic is permitted to flow without IPSec protection.

Deny Traffic is denied.

IpSec Traffic must be protected by IPSec. The IpFilterRule statement must also specify an IpManVpnAction statement or an IpDynVpnAction statement based on the type of tunnel (manual or dynamic) that is going to be used to provide IPSec protection for the traffic.

IpFilterLogging

Specifies a logging action that is applied to one or more filter rules (those that reference the IpGenericFilterAction statement). The logging action can be disabled by the setting of the FilterLogging parameter on the IpFilterPolicy statement.

IpFilterLogging (for IpFilterAction Permit or Deny)
Indicates whether a log record should be written when a packet matches this rule.

IpFilterLogging (for IpFilterAction IpSec)

No Log record is not written when a packet matches this rule.

Yes Log record is written when a packet matches this rule regardless of whether a valid SA is found or not.

LogPermit
Log record is written when a packet matches this rule and a valid SA is found.

LogDeny
Log record is written when a packet matches this rule and a valid SA is not found.

DiscardAction

Specifies a discard action that is applied to one or more filter rules (those that reference the IpGenericFilterAction statement). The discard action is applied whenever a packet is discarded. A packet might be discarded because the value deny is specified on the IpFilterAction parameter, but it might also be discarded for having a mismatch with filter policy (for example, a packet arrived over the wrong tunnel, or was sent in the clear when a tunnel was required).

Silent Specify this value to discard the packet silently.

ICMP Specify this value to send an ICMP or ICMPv6 destination unreachable error with reason `administratively prohibited` to the originating address of the discarded packet. ICMP errors are not generated for locally originated traffic; they are generated only for remote traffic that is being received or forwarded.

Guideline: If you specify ImplicitDiscardAction ICMP, create a filter rule permitting these ICMP errors.

Restriction: This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for more information.

IpLocalStartAction statement

Use the IpLocalStartAction statement to indicate how to determine the local IP, remote IP, local port, remote port, protocol specification, ICMP type and code specifications, and mobility header type specification for the local activation of a dynamic VPN. It provides information about the remote and local security endpoints with which dynamic SAs should be negotiated.

The IpLocalStartAction is optional for host-to-host dynamic SAs that are initiated locally. If this action is not specified, default values are used to locate a matching KeyExchangeRule keyword. The KeyExchangeRule keyword is searched based on the local and remote dynamic SA endpoints to be negotiated. If the IpLocalStartAction is not specified on the IpFilterRule statement, the remote IP security endpoint is supplied based on the destination IP address in an outbound packet in the case of an OnDemand request, or the RemoteIp keyword value in the case of activation based on a LocalDynVpnRule. The local IP security endpoint is supplied based on the source IP address in an outbound packet, or the LocalIp keyword value in the case of activation based on the LocalDynVpnRule statement.

If the IpLocalStartAction statement is not specified, the AllowOnDemand default policy specified on the IpFilterPolicy is used to determine whether OnDemand requests are allowed. Additionally, defaults for granularity of locally initiated SAs are determined as follows:

- The IP addresses used for the security endpoints are determined based on the outbound packet (OnDemand) or the LocalIp and RemoteIp keywords from the LocalDynVpnRule statement.
- The negotiated SA is based on the protocol value specified in the rule which can either be a specific protocol or all protocols.
- For both source port and destination port, if the matching filter rule specifies a single port value or all ports, the SA is negotiated with the port value from the rule. IKE version 1 negotiation can be done only with a single port or all ports. When the rule specifies a port range and IKE version 1 is used, the negotiation is done with the port specification from the outbound packet or the LocalDynVpnRule statement. When the rule specifies a port range and IKE version 2 is used, the negotiation is done with the port range specification.
- If the filter rule specifies an ICMP type and code, ICMPv6 type and code, or mobility header type, the negotiated SA is based on those specifications. IKE version 1 negotiation can only be done with ICMP type and code, ICMPv6 type and code, or mobility header type specification of any.

Syntax

```
►►──IpLocalStartAction──name──┤ Put Braces and Parameters on Separate Lines ├───────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬──────────────────────────────────────┤
   ├─┤ IpLocalStartAction Parameters ├──┤
   └─}──────────────────────────────────┘
```

IpLocalStartAction Parameters:

```
   ┌─AllowOnDemand ──No───────┐  ┌─LocalPortGranularity ──Rule─────────┐
├──┼──────────────────────────┼──┼─────────────────────────────────────┼──────────►
   └─AllowOnDemand──┬─Yes─┬───┘  └─LocalPortGranularity──┬─Rule───┬────┘
                    └─No──┘                              └─Packet─┘

   ┌─RemotePortGranularity ──Rule──────┐  ┌─ProtocolGranularity ──Rule───────┐
►──┼───────────────────────────────────┼──┼──────────────────────────────────┼─────►
   └─RemotePortGranularity──┬─Rule───┬─┘  └─ProtocolGranularity──┬─Rule───┬──┘
                            └─Packet─┘                           └─Packet─┘

   ┌─ICMPCodeGranularity ──Rule────────┐  ┌─ICMPTypeGranularity ──Rule───────┐
►──┼───────────────────────────────────┼──┼──────────────────────────────────┼─────►
   └─ICMPCodeGranularity──┬─Rule───┬───┘  └─ICMPTypeGranularity──┬─Rule───┬──┘
                          └─Packet─┘                             └─Packet─┘

   ┌─ICMPv6CodeGranularity ──Rule──────┐  ┌─ICMPv6TypeGranularity ──Rule─────┐
►──┼───────────────────────────────────┼──┼──────────────────────────────────┼─────►
   └─ICMPv6CodeGranularity──┬─Rule───┬─┘  └─ICMPv6TypeGranularity──┬─Rule───┬┘
                            └─Packet─┘                             └─Packet─┘

   ┌─MIPv6TypeGranularity ──Rule───────┐  ┌─RemoteIpGranularity ──Packet─────┐
►──┼───────────────────────────────────┼──┼──────────────────────────────────┼─────►
   └─MIPv6TypeGranularity──┬─Rule───┬──┘  └─RemoteIpGranularity──┬─Rule───┬──┘
                           └─Packet─┘                            └─Packet─┘

   ┌─LocalIpGranularity ──Packet───────┐
►──┼───────────────────────────────────┼──┬──────────────────────────────────┬─────►
   └─LocalIpGranularity──┬─Rule───┬────┘  ├─LocalSecuritySecurityEndpoint────┤
                         └─Packet─┘       └─LocalSecurityEndpointRef name────┘

►──┬──────────────────────────────────┬────────────────────────────────────────────►
   ├─RemoteSecurityEndpoint───────────┤
   └─RemoteSecurityEndpointRef name───┘

►──┬─────────────────────────────────────────────────┬──────────────────────────┤
   ├─InitiateToLocation──┬─IpAddr ipaddress─┬────────┤
   │                     └─Dns dnsname──────┘        │
   └─InitiateToLocationRef name──────────────────────┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IpLocalStartAction statement. The name cannot start with a dash (-) or contain any commas (,).

AllowOnDemand
Indicates whether outbound IP packets can result in an on demand activation of a phase 2 negotiation. The default of **No** disallows on-demand activations.

LocalIpGranularity
The LocalIpGranularity value is consulted only when creating an on-demand dynamic VPN. It specifies which of the following IP addresses should be used as the local IP address during a phase 2 negotiation:

- The source IP address specification of the matching IP filter rule
- The source IP address in the IP packet that resulted in the on-demand activation

RemoteIpGranularity
The RemoteIpGranularity is consulted only when creating an on-demand dynamic VPN. It specifies which of the following IP addresses should be used as the remote IP address during a phase 2 negotiation:

- The destination IP address specification of the matching IP filter rule
- The destination IP address in the IP packet that resulted in the on-demand activation

LocalPortGranularity
Specifies which of the following port values should be used as the local port specification during a phase 2 negotiation:

- The source port specification of the matching IP filter rule.
- The source port specification in the IP packet that resulted in the on-demand activation.

Restriction: If the matching IP filter rule has an IpService statement that specifies a local port range and the dynamic VPN is negotiated using IKE version 1, then the source port from the IP packet is used. IKE version 1 does not support port ranges for this purpose.

Tip: IKE version 1 does not support negotiating a single SA for a port range other than **All** ports. When using IKE version 1, if you want to negotiate a single phase 2 SA to cover all ports for local activations, then you must code a port specification of **All** on your IpService statement, in addition to a LocalPortGranularity of **Rule**.

RemotePortGranularity
Specifies which of the following port values should be used as the remote port specification during a phase 2 negotiation:

- The destination port specification of the matching IP filter rule.
- The destination port specification in the IP packet that resulted in the on-demand activation.

Restriction: If the matching IP filter rule has an IpService statement that specifies a destination port range and the dynamic VPN is negotiated using IKE version 1, then the destination port from the IP packet is used. IKE version 1 does not support port ranges for this purpose.

Tip: IKE version 1 does not support negotiating a single SA for a port range other than **All** ports. When using IKE version 1, if you want to negotiate a single phase 2 SA to cover all ports for local activations, then you must code a port specification of **All** on your IpService statement, in addition to a RemotePortGranularity of **Rule**.

ProtocolGranularity
Specifies which of the following protocol values should be used as the protocol specification during a phase 2 negotiation:

- The protocol specification of the matching IP filter rule
- The protocol specification in the IP packet that resulted in the on-demand activation

ICMPCodeGranularity
Specifies which of the following ICMP code values should be used during an IKE version 2 phase 2 negotiation:

- The ICMP code specification of the matching IP filter rule
- The ICMP code specification in the IP packet that resulted in the on-demand activation

The ICMPCodeGranularity parameter is ignored when IKE version 1 is used. The ICMP code specification of the matching IP filter rule is used during an IKE version 1 phase 2 negotiation. An IKE version 1 negotiation fails if the matching IP filter rule ICMP code specification contains a value other than any.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

ICMPTypeGranularity
Specifies which of the following ICMP type values should be used during an IKE version 2 phase 2 negotiation:

- The ICMP type specification of the matching IP filter rule
- The ICMP type specification in the IP packet that resulted in the on-demand activation

The ICMPTypeGranularity parameter is ignored when IKE version 1 is used. The ICMP type specification of the matching IP filter rule is used during an IKE version 1 phase 2 negotiation. An IKE version 1 negotiation fails if the matching IP filter rule ICMP type specification contains a value other than any.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

ICMPv6CodeGranularity
Specifies which of the following ICMPv6 code values should be used during an IKE version 2 phase 2 negotiation:

- The ICMPv6 code specification of the matching IP filter rule
- The ICMPv6 code specification in the IP packet that resulted in the on-demand activation

The ICMPv6CodeGranularity parameter is ignored when IKE version 1 is used. The ICMPv6 code specification of the matching IP filter rule is used during an IKE version 1 phase 2 negotiation. An IKE version 1 negotiation fails if the matching IP filter rule ICMPv6 code specification contains a value other than any.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

ICMPv6TypeGranularity
Specifies which of the following ICMPv6 type values should be used during an IKE version 2 phase 2 negotiation:

- The ICMPv6 type specification of the matching IP filter rule

- The ICMPv6 type specification in the IP packet that resulted in the on-demand activation

The ICMPv6TypeGranularity parameter is ignored when IKE version 1 is used. The ICMPv6 type specification of the matching IP filter rule is used during an IKE version 1 phase 2 negotiation. An IKE version 1 negotiation fails if the matching IP filter rule ICMPv6 type specification contains a value other than any.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

`MIPv6TypeGranularity`
Specifies which of the following mobility header type values should be used during an IKE version 2 phase 2 negotiation:

- The MIPv6 type specification of the matching IP filter rule
- The MIPv6 type specification in the IP packet that resulted in the on-demand activation

The MIPv6TypeGranularity parameter is ignored when IKE version 1 is used. The MIPv6 type specification of the matching IP filter rule is used during an IKE version 1 phase 2 negotiation. An IKE version 1 negotiation fails if the matching IP filter rule MIPv6 type specification contains a value other than any.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

`LocalSecurityEndpoint`
An inline specification of a LocalSecurityEndpoint statement.

The LocalSecurityEndpoint statement is used to locate a KeyExchangeRule statement that indicates how IKE negotiations are to be protected.

The LocalSecurityEndpoint statement is optional for host-to-host and host-to-gateway configurations. If this statement is not specified, default values are used to locate a matching KeyExchangeRule statement. The KeyExchangeRule statement is located based on the local and remote dynamic SA endpoints to be negotiated. The local IP security endpoint is supplied based on the source IP address in an outbound packet in the case of an on-demand activation or the LocalIp keyword in the case of activation based on a LocalDynVpnRule statement.

`LocalSecurityEndpointRef`
The name of a globally defined LocalSecurityEndpoint statement. The LocalSecurityEndpoint statement is used to locate a KeyExchangeRule statement that indicates how IKE negotiations are to be protected.

`RemoteSecurityEndpoint`
An inline specification of an RemoteSecurityEndpoint statement. The RemoteSecurityEndpoint statement is used to locate a KeyExchangeRule statement that indicates how IKE negotiations are to be protected.

`RemoteSecurityEndpointRef`
The name of a globally defined RemoteSecurityEndpoint statement. The RemoteSecurityEndpoint statement is used to locate a KeyExchangeRule statement that indicates how IKE negotiations are to be protected.

`InitiateToLocation`

IpAddr
: The IP address specification of the remote security endpoint to be used when initiating a dynamic VPN tunnel.

Dns The DNS name of the remote security endpoint to be used when initiating a dynamic VPN tunnel. The maximum length of DNS name is 512.

The InitiateToLocation parameter is optional for host-to-host or gateway-to-host configurations. If the parameter is not specified, the InitiateToLocation parameter is determined at run time. For on-demand activations, the destination address in the IP packet that triggered the activation is used. For activations based on a LocalDynVpnRule statement, the IP address from the RemoteIP keyword is used. The IP Address specified for InitiateToLocation should be included within the subnet or range of IP addresses specified on the RemoteSecurityEndpoint parameter location. If the RemoteSecurityEndpoint parameter specifies a single IP address for location, the InitiateToLocation parameter should match the RemoteSecurityEndpoint parameters location value.

`InitiateToLocationRef`
The name of a globally defined IpAddr statement for the remote security endpoint to be used when initiating a dynamic VPN tunnel.

Rules:

- All Location addresses in LocalSecurityEndpoint and RemoteSecurityEndpoint for this action must be in the same address family (IPv4 or IPv6).
- The address for the IpFilterRule statement associated with this action must be in the same address family as the Location addresses in the LocalSecurityEndpoint and RemoteSecurityEndpoint parameters.

IpManVpnAction statement

Use the IpManVpnAction statement to indicate how selected traffic between two security endpoints should be protected utilizing manually established security associations. An IpTimeCondition statement can be used to identify when the manual tunnel is installed in the stack. Activation of the manual tunnel is controlled by the Active parameter and the **`ipsec`** command activate/deactivate function.

Syntax

```
►►──IpManVpnAction──name──┤ Put Braces and Parameters on Separate Lines ├──────────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬────────────────────────────────────────────────────────┤
   ├─┤ IpManVpnAction Parameters ├─┤
   └─}──────────────────────────┘
```

IpManVpnAction Parameters:

```
   ┌─Active Yes─────────┐
├──┼────────────────────┼──┬─LocalSecurityEndpointAddr──┬─address─┬─┬──────────────────────────►
   └─Active──┬─Yes─┬────┘  │                            ├─Any─────┤ │
             └─No──┘       │                            ├─Any4────┤ │
                           │                            └─Any6────┘ │
                           └─LocalSecurityEndpointAddrRef name──────┘

                                                   ┌─PassthroughDF Yes─────────────────┐
►──┬─RemoteSecurityEndpointAddr──┬─address─┬─┬─────┼───────────────────────────────────┼──────►
   │                             ├─Any─────┤ │     └─PassthroughDF──┬─No──┬─Clear─┬──┬─┘
   │                             ├─Any4────┤ │                      │     │       │  │
   │                             └─Any6────┘ │                      │     ├─Set───┤  │
   └─RemoteSecurityEndpointAddrRef name──────┘                      │     └─Clear─┘  │
                                                                    └─Yes────────────┘

   ┌─PassthroughDSCP Yes──────┐
►──┼──────────────────────────┼────────────────────────────────────────────────────────────────►
   └─PassthroughDSCP──┬─No──┬─┘
                      └─Yes─┘

►─HowToAuth──┬─AH──┬──┬─AES128_XCBC_96──────┬─┤ AuthOutboundSa ├─┤ AuthInboundSa ├───────────►
             └─ESP─┘  ├─HMAC_MD5────────────┤
                      ├─HMAC_SHA────────────┤
                      ├─HMAC_SHA1───────────┤
                      ├─HMAC_SHA2_256_128───┤
                      ├─HMAC_SHA2_384_192───┤
                      └─HMAC_SHA2_512_256───┘

►──┬──────────────────────────────────────────────────────────────────────────────────────┬──►
   └─HowToEncrypt──┬─DES──────────────────────┬─┤ EncryptOutboundSa ├─┤ EncryptInboundSa ├─┘
                   ├─3DES─────────────────────┤
                   ├─AES──────────────────────┤
                   └─AES_CBC KeyLength keylen─┘

                                   ┌──────────────────────────┐
                                   ▼                          │
►─HowToEncap──┬─Tunnel────┬─────┬─────────────────────────┬───┴──────────────────────────────┤
              └─Transport─┘     ├─IpTimeCondition─────────┤
                                └─IpTimeConditionRef name─┘
```

AuthOutboundSa:

```
├──AuthOutboundSa──spi──key──────────────────────────────────────────────────────────────────┤
```

AuthInboundSa:

```
├──AuthInboundSa──spi──key───────────────────────────────────────────────────────────────────┤
```

EncryptOutboundSa:

```
├──EncryptOutboundSa──spi──key───────────────────────────────────────────────────────────────┤
```

EncryptInboundSa:

```
├──EncryptInboundSa──spi──key──────────────────────────────────────────────────┤
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IpManVpnAction statement. The name cannot start with a dash (-) or contain any commas (,).

Active
: An indication of whether the tunnel state is set to active or inactive when the manual tunnel is installed in the stack. If a Active value of **No** is specified, then the **ipsec** command must be used to activate the manual tunnel.

  Results:

  - If Active **Yes** is specified (default), the IpManVpnAction statement is activated automatically when the policy is installed. If an IpTimeCondition is present on the action, that controls when the policy is installed.
  - If Active **No** is specified, the IpManVpnAction statement must be manually activated using the **ipsec** command before it can be used to protect IP traffic. IP packets matching on the associated IpFilterRule are dropped until the IpManVpnAction statement is activated.

LocalSecurityEndpointAddr *name*

address	The IP address of the local security endpoint. **Restriction:** The IPv6 unspecified address (::0) and IPv4 unspecified address (0.0.0.0) are not allowed.
Any	Indicates that any local IPv4 address can be used for the local security endpoint. Any and Any4 are interchangeable values. **Restriction:** This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.
Any4	Indicates that any local IPv4 address can be used for the local security endpoint. **Restriction:** This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.
Any6	Indicates that any local IPv6 address can be used for the local security endpoint.

LocalSecurityEndpointAddrRef
: The name of a globally defined IpAddr statement for the local security endpoint.

RemoteSecurityEndpointAddr

address	The IP address of the remote security endpoint. **Restriction:** The IPv6 unspecified address (::0) and IPv4 unspecified address (0.0.0.0) are not allowed.
Any	Indicates that any remote IPv4 address can be used for the remote security endpoint. Any and Any4 are interchangeable values. **Restriction:** This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

Any4 Indicates that any remote IPv4 address can be used for the remote security endpoint.

Restriction: This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Any6 Indicates that any remote IPv6 address can be used for the remote security endpoint.

RemoteSecurityEndpointAddrRef *name*
The name of a globally defined IpAddr statement for the remote security endpoint.

PassthroughDF
When this value is set to No, the do not fragment bit is set to 0 (if the value Clear is specified) or 1 (if the value Set is specified) on the outer IP header for an IPv4 tunnel mode SA. When this value is set to Yes, the do not fragment bit is copied from the inner IP header to the outer IP header for an IPv4 tunnel mode SA. This setting is ignored for IPv6 or transport mode SAs.

Restriction: This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

PassthroughDSCP
When this value is set to No, the Differentiated Services Code Point (DSCP) field is set to 0 on the outer IP header for a tunnel mode SA. When this value is set to Yes, the DSCP field is copied from the inner IP header to the outer IP header for a tunnel mode SA. This setting is ignored for transport mode SAs.

Restriction: This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

HowToAuth
The authentication protocol and algorithm used to provide data integrity. The following protocols can be specified.

AH Use AH headers to carry authentication data.

ESP Use ESP headers to carry authentication data.

The following algorithms can be specified. The algorithms are ordered from least to most secure.

HMAC_MD5
Computes the authentication checksum by combining a 128–bit key, the Hash-based Message Authentication Code (HMAC) authentication algorithm and the MD5 hash algorithm.

Restriction: HMAC_MD5 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

AES128_XCBC_96
Computes the authentication checksum using the AES128_XCBC keyed hash algorithm with a 128-bit key and a 96-bit Integrity Check Value (ICV).

Restriction: AES128_XCBC_96 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HMAC_SHA
Deprecated and treated as a synonym for HMAC_SHA1.

HMAC_SHA1
Computes the authentication checksum by combining a 160–bit key, the HMAC authentication algorithm and the Secure Hash Algorithm (SHA) hash algorithm.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HMAC_SHA2_256_128
Computes the authentication checksum using the HMAC_SHA2_256 keyed hash algorithm with a 256-bit key and 128-bit ICV.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HMAC_SHA2_384_192
Computes the authentication checksum using the HMAC_SHA2_384 keyed hash algorithm with a 384-bit key and a 192-bit ICV.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HMAC_SHA2_512_256
Computes the authentication checksum using the HMAC_SHA2_512 keyed hash algorithm with a 512-bit key and a 256-bit ICV.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

AuthOutboundSa
Specifies the SA parameters for authentication traffic transmitted outbound to the remote security endpoint.

spi Specifies the remote Security Parameter Index. Valid values for *spi* are in the range 1 - 4 294 967 294. The set of SPI values in the range 1 - 255 are reserved to the Internet Assigned Numbers Authority (IANA) for future use.

key Specifies the authentication key. The key must be specified in hexadecimal prefixed with '0x'. Each byte of the key represents a value in the range 00 - FF. The length of the key is determined by the associated algorithm. The key length (in bytes) for each algorithm type is:

- HMAC_MD5 (16)
- AES128_XCBC_96 (16)
- HMAC_SHA1 (20)
- HMAC_SHA2_256_128 (32)
- HMAC_SHA2_384_192 (48)
- HMAC_SHA2_512_256 (64)

AuthInboundSa
Specifies the SA parameters for authentication traffic received inbound from the remote security endpoint.

spi Specifies the local Security Parameter Index. Valid values for *spi* are in the range 1 - 4 294 967 294.

Guidelines:

- The set of SPI values in the range 1 - 255 is reserved to the Internet Assigned Numbers Authority (IANA) for future use.

- Consider choosing an inbound SPI value in the range 256 - 4096. These values are reserved by TCP/IP for use by manual tunnels and do not conflict with any dynamic tunnels.

key Specifies the authentication key. The key must be specified in hexadecimal prefixed with ′0x′. Each byte of the key represents a value in the range 00-FF. The length of the key is determined by the associated algorithm. The key length (in bytes) for each algorithm type is:

- HMAC_MD5 (16)
- AES128_XCBC_96 (16)
- HMAC_SHA1 (20)
- HMAC_SHA2_256_128 (32)
- HMAC_SHA2_384_192 (48)
- HMAC_SHA2_512_256 (64)

HowToEncrypt

Encryption is done using the ESP protocol. Specify the encryption algorithm used to provide data confidentiality. The algorithms are ordered from least to most secure.

DES DES encryption is used with a 56–bit key and a 64–bit initialization vector.

Restriction: DES is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

3DES Triple DES runs the DES encryption algorithm three times and uses 192-bits, including 24 parity bits.

Rule: If 3DES is specified but is not supported by the system, then the Policy Agent fails the policy.

AES Deprecated and treated as a synonym for AES_CBC KeyLength 128.

Rule: If AES is specified but AES encryption in CBC mode is not supported by TCP/IP, Policy Agent fails the policy.

AES_CBC KeyLength *keylen*

The AES algorithm is used in Cipher Block Chaining (CBC) mode with a key length *length*, either 128 or 256 bits.

Rule: If AES_CBC is specified but AES encryption in CBC mode is not supported by TCP/IP, Policy Agent fails the policy.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

EncryptOutboundSa

Specifies the SA parameters for encryption traffic transmitted outbound to the remote security endpoint.

spi Specifies the remote Security Parameter Index. Valid values for *spi* are in the range 1 - 4 294 967 294. The set of SPI values in the range 1 - 255 are reserved to the Internet Assigned Numbers Authority (IANA) for future use.

key Specifies the encryption key. The key must be specified in hexadecimal prefixed with ′0x′. Each byte of the key represents a value 00-FF. The length of the key is determined by the associated algorithm. The key length (in bytes) for each algorithm type is:

- DES (8)
- 3DES_CBC (24)
- AES_CBC KeyLength 128 (16)
- AES_CBC KeyLength 256 (32)

`EncryptInboundSa`

Specifies the SA parameters for encryption traffic received inbound from the remote security endpoint.

spi Specifies the local Security Parameter Index. Valid values for *spi* are in the range 1 - 4 294 967 294.

Guidelines:

- The set of SPI values in the range 1 - 255 is reserved to the Internet Assigned Numbers Authority (IANA) for future use.
- Consider choosing an inbound SPI value in the range 256 - 4 096. These values are reserved by TCP/IP for use by manual tunnels and do not conflict with any dynamic tunnels.

key Specifies the encryption key. The key must be specified in hexadecimal prefixed with '0x'. Each byte of the key represents a value in the range 00 - FF. The length of the key is determined by the associated algorithm. The key length (in bytes) for each algorithm type is:

- DES (8)
- 3DES_CBC (24)
- AES_CBC KeyLength 128 (16)
- AES_CBC KeyLength 256 (32)

`HowToEncap`

An indication of whether IPSec-protected packets should be created using tunnel mode encapsulation or transport mode encapsulation.

Transport mode provides protection for the transport-layer headers and data (for example, TCP or UDP packet) inside an IP packet. This mode is used when the endpoints of the secure tunnel are the two communicating systems.

Tunnel mode provides protection for the entire IP packet. This mode is usually used for a secure tunnel between two gateways or between a gateway and a remote system.

`IpTimeCondition`

An inline specification of an IpTimeCondition statement. There is a limit of 25 IpTimeCondition specifications and references on the IpManVpnAction statement.

`IpTimeConditionRef`

The name of a globally defined IpTimeCondition statement. There is a limit of 25 IpTimeCondition specifications and references on the IpManVpnAction statement.

Rules:

- If ESP authentication is being used with encryption, the SPI values on the EncryptInboundSa and AuthInboundSa parameters must be the same value. Also, the SPI values on EncryptOutboundSa and AuthOutboundSa parameters must be the same value.

- The combination of inbound SPI value, LocalSecurityEndpointAddr, and RemoteSecurityEndpointAddr that you specify for ESP encapsulation must be unique across the entire set of IpManVpnAction statements. The following values are ESP encapsulation SPI values:
  - SPI value specified on the EncryptInboundSa parameter
  - SPI value specified on the AuthInboundSa parameter, if HowToAuth ESP is specified
- The combination of inbound SPI value, LocalSecurityEndpointAddr, and RemoteSecurityEndpointAddr that you specify for AH encapsulation must be unique across the entire set of IpManVpnAction statements. The following value is the AH encapsulation SPI value:
  - SPI value specified on the AuthInboundSa parameter, if HowToAuth AH is specified
- If ESP authentication is being used without encryption, the ESP header is present, but the payload is not encrypted (ESP_NULL).
- Replay prevention is not supported for manual security associations.
- All IpManVpnAction addresses must be in the same address family (IPv4 or IPv6).
- The addresses for the IpFilterRule statement associated with this action must be in the same address family as the addresses for this action.

Results:

- The setting of the Active parameter is applied each time the manual tunnel is installed in the stack. A change to any parameter on the IpManVpnAction statement (including the Active parameter) results in the manual tunnel being reinstalled in the stack and the Active parameter being applied. For example, in the case where the Active parameter is set to **No** and the manual tunnel has been activated with the **ipsec -m activate** command, a change to the encryption key results in the tunnel being reinstalled and the state being set to inactive.
- If both HowToAuth and HowToEncrypt are specified, the semantic is that encryption is always applied to the payload before authentication.
- If you specify Any, Any4, or Any6 for the LocalSecurityEndpointAddr or RemoteSecurityEndpointAddr parameters, and you set HowToEncap to Transport, then encapsulation preserves the original source or destination address in the IP header.
- If you specify Any, Any4, or Any6 for the LocalSecurityEndpointAddr or RemoteSecurityEndpointAddr parameters, and you set HowToEncap to Tunnel, then encapsulation preserves the original source or destination address in the IP header, if possible. If necessary, the source address is changed to an appropriate source address on the local stack.

Tips:

- Use the **`ipsec`** command to activate and deactivate manual tunnels.
- Manual tunnels must be activated at both security endpoints. Unlike dynamic tunnels, there is no responder mode activation for manual tunnels.
- Because multicast traffic is one-to-many but can be used both for sending and receiving, using manual tunnels for multicast requires the same SPI and keys for inbound and outbound traffic.

IpService statement

Use the IpService statement to provide a coupling between IP transport conditions, IP routing conditions, and actions.

Syntax

```
►►──IpService──┬──────┬──┤ Put Braces and Parameter on Separate Lines ├──────────►◄
               └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────┬─────────────────────────────────────────────────┤
   ├─┤ IpService Parameters ├──┤
   └─}─────────────────────────┘
```

IpService Parameters:

```
   ┌─Protocol All───────────────────────────────────────────────┐
├──┴─Protocol──┬─┬─Tcp─┬──┤ PortSpecification ├─────────────┬──┴──────────────────►
               │ └─6───┘                                    │
               ├─┬─Udp─┬──┤ PortSpecification ├─────────────┤
               │ └─17──┘                                    │
               ├─┬─Icmp─┬──┤ IcmpSpecification ├────────────┤
               │ └─1────┘                                   │
               ├─┬─Icmpv6─┬──┤ IcmpSpecification ├──────────┤
               │ └─58─────┘                                 │
               ├─┬─Ospf─┬──┤ OspfSpecification ├────────────┤
               │ └─89───┘                                   │
               ├─┬─MIPv6─┬──┤ MIPv6Specification ├─────────┤
               │ └─135───┘                                  │
               ├─┬─Ip─┬─────────────────────────────────────┤
               │ └─4──┘                                     │
               ├─┬─Ipip─┬───────────────────────────────────┤
               │ └─94───┘                                   │
               ├─Ah─────────────────────────────────────────┤
               ├─Esp────────────────────────────────────────┤
               ├─Igmp───────────────────────────────────────┤
               ├─All────────────────────────────────────────┤
               ├─Opaque─────────────────────────────────────┤
               └─n──────────────────────────────────────────┘

►──Direction──┬─Inbound──────────────────────────────────┬────────────────────────►
              ├─Outbound─────────────────────────────────┤
              └─Bidirectional──┬───────────────────┬─────┘
                               ├─InboundConnect────┤
                               └─OutboundConnect───┘

                                                        ┌─SecurityClass 0─┐
►──Routing──┬─Local──────────────────────────────────┬──┴─SecurityClass n─┴──────┤
            ├─Routed──┤ FragmentSpecification ├──────┤
            └─Either─────────────────────────────────┘
```

PortSpecification:

```
   ┌─SourcePortRange 0───────────┐ ┌─DestinationPortRange 0──────────┐
├──┴─SourcePortRange──┬─n─────┬──┴─┴─DestinationPortRange ──┬─n─────┬─┴──────────────┤
                      └─n m───┘                             └─n m───┘
```

IcmpSpecification:

```
   ┌─Type Any───────┐ ┌─Code Any───────┐
├──┴─Type──┬─Any──┬─┴─┴─Code──┬─Any──┬─┴───────────────────────────────┤
           ├─n────┤           ├─n────┤
           └─n m──┘           └─n m──┘
```

MIPv6Specification:

```
   ┌─Type Any───────┐
├──┴─Type──┬─Any──┬─┴────────────────────────────────────────────────────┤
           ├─n────┤
           └─n m──┘
```

OspfSpecification:

```
   ┌─Type Any───────┐
├──┴─Type──┬─Any──┬─┴────────────────────────────────────────────────────┤
           └─n────┘
```

FragmentSpecification:

```
   ┌─FragmentsOnly No─────────┐
├──┴─FragmentsOnly──┬─No──┬───┴──────────────────────────────────────────┤
                    └─Yes─┘
```

Parameters

name

A string 1 - 32 characters in length specifying the name of this IpService statement.

Rule: If this IpService statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IpService, a nonpersistent system name is created.

Protocol

Indicates the protocol that must be contained in an IP packet for this rule's action to be performed. If an *n* value is specified it identifies a protocol number. The value for *n* can be in the range 0 - 255. If a value of **All** is specified, then the rule applies to any protocol.

The value Opaque matches any IPv6 packet for which the upper-layer protocol is not known as a result of fragmentation. This parameter always matches non-initial fragments, and it also matches initial fragments if the upper-layer protocol value is not included in the first fragment. The Opaque value is applicable only to routed fragments because, for all local traffic, the stack applies IP filter rules only to fully assembled packets.

The protocol name **Ip** maps to the value 4, representing IP in IP encapsulation, for which IANA has assigned the name IP.

The name **Ipip** maps to the value 94, representing IP within IP encapsulation, for which IANA has assigned the name IPIP.

Restriction: The values MIPv6 and Opaque are valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

SourcePortRange
If a Protocol of TCP or UDP is specified, then a SourcePortRange value can be specified. The SourcePortRange value indicates the applicable source ports that must be contained in an IP packet for this rule's action to be performed.

Valid values for *n* are 0 - 65 535. If 0 is specified for *n*, then the rule applies to any source port. If *n* is specified as the beginning value for a range, then 0 is not a valid value.

If an *m* value is specified, it must be greater than or equal to *n* and less than 65 536.

DestinationPortRange
If a Protocol of TCP or UDP is specified, then a DestinationPortRange value can be specified. The DestinationPortRange value indicates the applicable destination ports that can be contained in an IP packet for this rule's action to be performed.

Valid values for *n* are in the range 0 - 65 535. If 0 is specified for *n*, then the rule applies to any destination port. If *n* is specified as the beginning value for a range, then 0 is not a valid value.

If an *m* value is specified, then it must be greater than or equal to *n* and less than 65 536.

Type
If you specify Protocol ICMP or ICMPv6, then you can specify a Type value or range. The Type value indicates the ICMP types that must be contained in an IP packet for this rule's action to be performed. Valid values for *n* are in the range 0 - 255. If you specify an *m* value, it must be greater than or equal to *n* and less than or equal to 255.

If you specify Protocol Ospf, then you can specify Type. The Type value indicates the OSPF types that must be contained in an IP packet for this rule's action to be performed. Valid values for *n* are in the range 0 - 255.

If you specify Protocol MIPv6, then you can specify a Type value or range. The Type value indicates the mobility header types that must be contained in an IP packet for this rule's action to be performed. Valid values for *n* are in the range 0 - 255. If you specify an *m* value, it must be greater than or equal to *n* and less than or equal to 255.

Restrictions:

- The use of a range of values for certain protocols is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.
- ICMP, ICMPv6 and Mobility header Type specifications other than Any are allowed for filter rules that reference an IpDynVpnAction statement, but it is valid only when the SA is negotiated using IKE version 2. Because the IKE

version is not determined until IKE negotiations begin, the IKE daemon fails an SA negotiation under such a rule if the chosen KeyExchangeRule calls for IKE version 1.

Code

If you specify Protocol ICMP or ICMPv6, then you can specify a Code value or range. The Code value indicates the ICMP codes that must be contained in an IP packet for this rule's action to be performed. Valid values for *n* are in the range 0 - 255. If an *m* value is specified, it must be greater than or equal to *n* and less than or equal to 255.

Restrictions:

- The use of a range of values for certain protocols is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.
- ICMP and ICMPv6 Code specifications other than Any are allowed for filter rules that reference an IpDynVpnAction statement, but it is valid only when the SA is negotiated using IKE version 2. Because the IKE version is not determined until IKE negotiations begin, the IKE daemon fails an SA negotiation under such a rule if the chosen KeyExchangeRule calls for IKE version 1.

Direction

Specifies the direction a packet must take in order for the generated IP filters to apply.

Outbound
This value generates one IP filter. The generated rule permits or denies a packet with the specified source and destination to travel outbound.

Inbound
This value generates one IP filter. The generated rule permits or denies a packet with the specified source and destination to travel inbound.

Bidirectional
This value generates two IP filters. The first generated rule permits or denies a packet with the specified source and destination IP address or port to travel outbound. The second generated rule switches the source and destination specifications and permits or denies a packet with the switched source and destination specification to travel inbound.

InboundConnect/OutboundConnect
When Bidirectional is specified for Direction, an additional InboundConnect or OutboundConnect keyword can also be specified. These values are ignored if the protocol is not TCP. InboundConnect or OutboundConnect controls the type of packet that can send the first packet of a TCP connection (for example, the type of packet that can initiate a TCP connection). If InboundConnect and Protocol TCP are specified, then a TCP connection can be initiated only by an inbound packet. If OutboundConnect and Protocol TCP are specified, then a TCP connection can be initiated only by an outbound packet.

Routing

Specifies the type of packet that applies to this rule.

Local Indicates that this rule applies to packets destined for this stack.

Routed
Indicates that this rule applies to packets being forwarded by this stack.

Either Indicates that this rule applies to forwarded and non-forwarded packets.

SecurityClass
An IP packet must traverse a physical interface with a SecurityClass value of *n* to match the generated rule. The interface security class is defined on the LINK, INTERFACE, or DYNAMICXCF statement in the TCP/IP profile. Valid values for *n* can be a value in the range 0 - 255. The value 0 indicates that any interface is allowed. The SecurityClass parameter must be specified as 0 if the IpService statement is referenced by an IpFilterRule statement that also references an IpDynVpnAction statement.

FragmentsOnly
When this parameter is set to Yes, this rule matches only fragmented packets. When this parameter is set to No, this rule matches both fragments and non-fragments.

Restriction: This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

Tip: Fragments are only matched in routed traffic, because the TCP/IP stack applies IP filter rules for local traffic only to fully reassembled packets.

Rule: An FragmentsOnly specification of Yes is not allowed for filter rules that reference an IpDynVpnAction statement.

Tip: To specify all ephemeral ports for the SourcePortRange or DestinationPortRange keywords, you can specify ports in the range 1 024 - 65 535.

Rules:

- Filter rules that reference an IpManVpnAction statement or IpDynVpnAction statement must have a Direction of Bidirectional specified on the IpService parameter.
- A Routing specification of Routed or Either must have one of the following:
  - A SourcePortRange and DestinationPortRange specification defaulted or configured to 0 (if applicable)
  - A Type and Code specification defaulted or configured to Any (if applicable)

  This restriction is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.
- Filter rules that reference an IpDynVpnAction must have a SecurityClass value of 0 specified on the IpService statement.
- An ICMP or ICMPv6 Type and Code specification other than Any is allowed for filter rules that reference an IpDynVpnAction statement but it is valid only when the SA is negotiated using IKE version 2. Because the IKE version is not determined until IKE negotiations begin, the IKE daemon fails an SA negotiation under such a rule if the chosen KeyExchangeRule calls for IKE version 1.
- The ICMP or ICMPv6 Code specification must be set to the Any value if a range of ICMP or ICMPv6 Types is specified.
- An OSPF Type specification is not allowed for filter rules that reference an IpDynVpnAction statement.
- A mobility header Type specification other than Any is allowed for filter rules that reference an IpDynVpnAction statement, but it is valid only when the SA is

negotiated using IKE version 2. Because the IKE version is not determined until IKE negotiations begin, the IKE daemon fails an SA negotiation under such a rule if the chosen KeyExchangeRule calls for IKE version 1.

- A protocol specification of Opaque can be used only in combination with IPv6 addresses on an IpFilterRule.
- A protocol specification of Opaque is not allowed for filter rules that reference an IpDynVpnAction statement.

IpServiceGroup statement

Use the IpServiceGroup statement to define an IP service group. An IpServiceGroup statement identifies a set of IpService statements that make up the IP service group. An IpServiceGroup statement can be referenced by an IpFilterRule statement. The IpServiceGroup statement is an advanced configuration feature that results in multiple IpFilterRules being generated if the group contains or references more than one IP service condition.

Syntax

```
►►──IpServiceGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{───────────────────────────┬──────────────────────────────────────┤
   │  ┌──────────────────────┐   │
   │  ▼                      │   │
   ├────┬─IPService──────────┴───┤
   │    └─IpServiceRef name─┘    │
   └─}───────────────────────────┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this IpServiceGroup statement.

IpService
An inline specification of an IpService statement to be included in this group.

IpServiceRef
The name of a globally defined IpService statement to be included in the group.

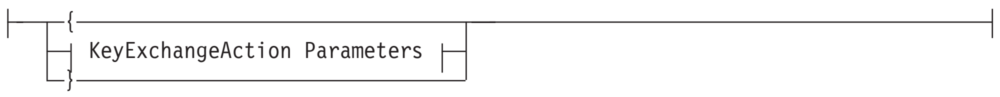
KeyExchangeAction statement

Use the KeyExchangeAction statement to define a key exchange action for a dynamic VPN. A key exchange indicates how key exchanges between the security endpoints should be protected. A KeyExchangeAction statement can be referenced by a KeyExchangeRule statement.

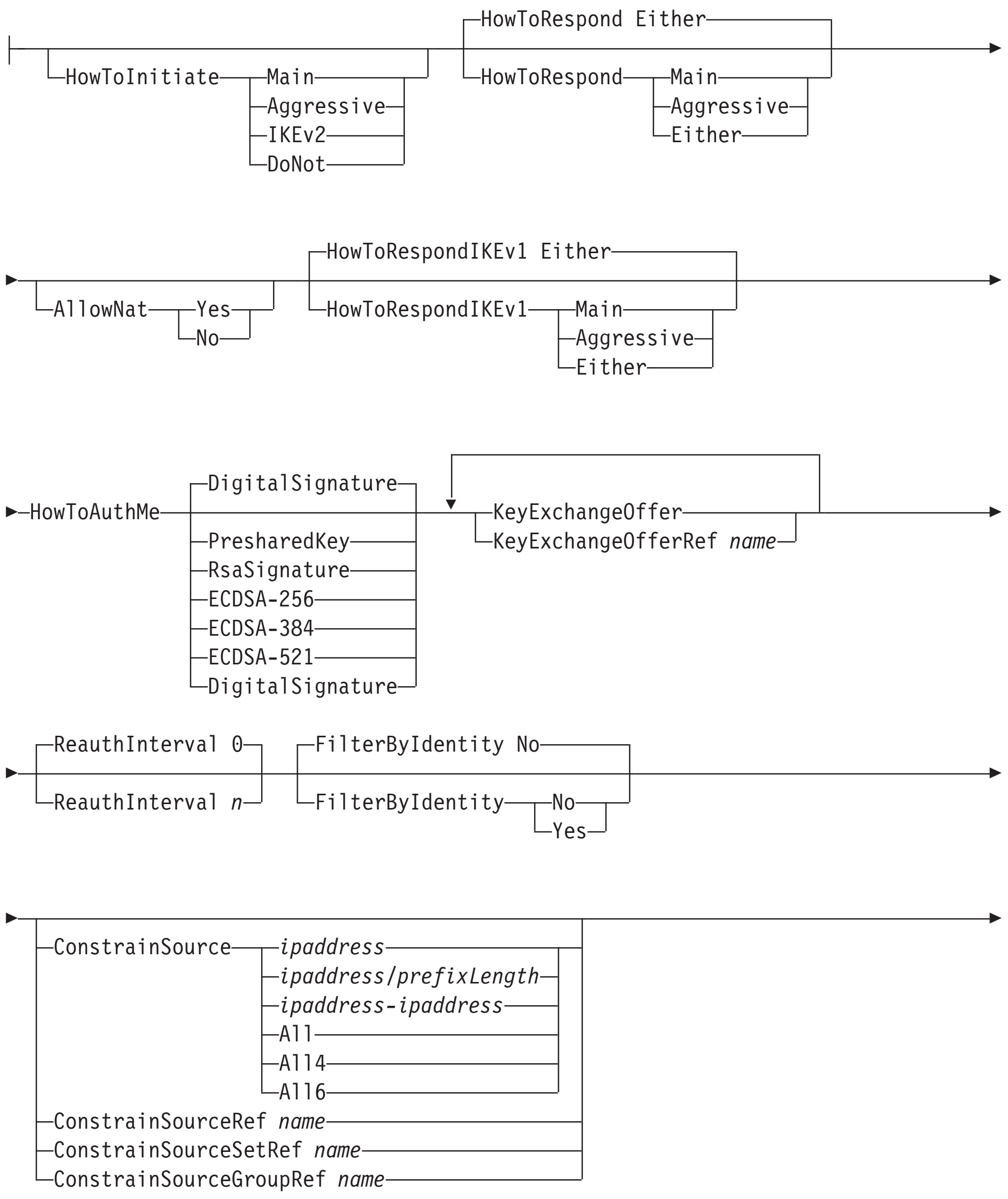
Syntax

```
►►──KeyExchangeAction──name──┤ Put Braces and Parameters on Separate Lines ├────────►◄
```

Put Braces and Parameters on Separate Lines:



KeyExchangeAction Parameters:



```
►─┬──────────────────────────────────────────────┬──────────────────────────────────────►
  ├─ConstrainDest──┬─ipaddress──────────────────┬─┤
  │                ├─ipaddress/prefixLength─────┤ │
  │                ├─ipaddress-ipaddress────────┤ │
  │                ├─All────────────────────────┤ │
  │                ├─All4───────────────────────┤ │
  │                └─All6───────────────────────┘ │
  ├─ConstrainDestRefname──────────────────────────┤
  ├─ConstrainDestSetRefname───────────────────────┤
  └─ConstrainDestGroupRefname─────────────────────┘
```

```
►─┬────────────────────────────────┬──────────────────────────────────────────────────────►
  └─BypassIpValidation──┬─Yes─┬────┘
                        └─No──┘
```

```
►─┬──────────────────────────────────────────────────────┬────────────────────────────────►
  └─CertificateURLLookupPreference──┬─Allow────┬─────────┘
                                    ├─Tolerate─┤
                                    └─Disallow─┘
```

```
►─┬─────────────────────────────────────┬────────────────────────────────────────────────┤
  └─RevocationChecking──┬─None────┬─────┘
                        ├─Loose───┤
                        └─Strict──┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this KeyExchangeAction statement. The name cannot start with a dash (-) or contain any commas (,).

HowToInitiate
: The negotiation mode to use as the phase 1 initiator. If this parameter is not specified, the IKE daemon will use the value from the HowToInitiate parameter in the KeyExchangePolicy.

Main Indicates that IKE version 1 with identity protection is used when key negotiations are initiated by this system.

Aggressive
Indicates that IKE version 1 without identity protection is used when key negotiations are initiated by this system.

IKEv2 Indicates that IKE version 2 is used when key negotiations are initiated by this system.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

DoNot
Indicates that the local system cannot initiate a key exchange negotiation.

HowToRespond
Deprecated and treated as a synonym for HowToRespondIKEv1.

HowToRespondIKEv1
The negotiation mode to assume as the IKE version 1 phase 1 responder. The default is Either.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Main Requires remote systems to initiate key negotiations using IKE version 1 with identity protection.

Aggressive
Requires remote systems to initiate key negotiations using IKE version 1 without identity protection.

Either Allows remote systems to initiate key exchange negotiations using IKE version 1 with or without identity protection.

Tip: The z/OS IKE daemon is always capable of responding with the IKE version 2 protocol. The HowToRespondIKEv1 parameter determines which IKE version 1 modes are allowed when z/OS is the responder.

HowToAuthMe
Specifies the method that remote security endpoints are to use to authenticate this security endpoint during IKE version 2 IKE_SA negotiation. If not specified, this value defaults to DigitalSignature.

PresharedKey
Indicates that the remote security endpoint is expected to authenticate this security endpoint with a pre-shared key.

RsaSignature
Indicates that the remote security endpoint is expected to authenticate this security endpoint with RSA signatures.

ECDSA-256
Indicates that the remote security endpoint is expected to authenticate this security endpoint using ECDSA with SHA-256 on the P-256 curve.

ECDSA-384
Indicates that the remote security endpoint is expected to authenticate this security endpoint using ECDSA with SHA-384 on the P-384 curve.

ECDSA-521
Indicates that the remote security endpoint is expected to authenticate this security endpoint using ECDSA with SHA-512 on the P-521 curve.

DigitalSignature
Indicates that the local security endpoint may use either RsaSignature, ECDSA-256, ECDSA-384 or ECDSA-521 when creating the digital signature for the remote security endpoint to verify. This is the default.

Restrictions:
- The HowToAuthMe keyword is ignored when IKE version 1 IKE SAs are negotiated because IKE version 1 requires that both security endpoints use the same authentication method.
- If PresharedKey is specified, the KeyExchangeRule that references the KeyExchangeAction must specify the SharedKey parameter.
- This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

AllowNat
Indicates whether the use of NAT traversal techniques is allowed when negotiating a phase 1 SA and subsequent phase 2 SAs that are using that phase 1 SA. The value Yes indicates that negotiations that use NAT traversal techniques are allowed. The value No indicates that negotiations that use NAT traversal techniques are not allowed. If the AllowNat parameter is specified, it overrides the AllowNat setting from the KeyExchangePolicy statement. If the AllowNat parameter is not specified, the AllowNat setting from the KeyExchangePolicy statement is used as the default.

Tip: Setting AllowNat to No prevents the IKE daemon from sending NAT payloads or processing received NAT payloads as part of the tunnel negotiation. In some cases, tunnels traversing one or more NATs can still be activated even when AllowNat is set to No. However, such tunnels are normally unusable because of the known incompatibilities between IPsec and NAT documented in RFC 3715.

KeyExchangeOffer
An inline specification of a KeyExchangeOffer statement.

Restriction: A KeyExchangeAction statement is limited to a maximum of 48 KeyExchangeOffer or KeyExchangeOfferRef statements.

KeyExchangeOfferRef
The name of a globally defined KeyExchangeOffer statement.

Restriction: A KeyExchangeAction statement is limited to a maximum of 48 KeyExchangeOffer or KeyExchangeOfferRef statements.

Rule: When you specify multiple KeyExchangeOffer parameters, configure the HowToInitiate parameter with the value Main to send multiple key exchange offers when a negotiation is initiated.

Result: When you specify multiple KeyExchangeOffer parameters, if the KeyExchangeAction parameter is configured with the value HowToInitiate Aggressive and contains multiple KeyExchangeOffer statements, the parameters of the first KeyExchangeOffer statement are used for initiating an Aggressive mode negotiation.

ReauthInterval
Specifies how often, in minutes, IKE version 2 peers reauthenticate themselves. Valid values are in the range 0-9999. The value 0 indicates that the endpoints should never reauthenticate. The default value is 0 (do not perform automatic reauthentication). Reauthentication renegotiates the keys for the IKE and reauthenticates the security endpoints. When IKE version 2 peers reauthenticate, the IKE SA and all associated child SAs must be terminated and renegotiated.

Restriction: The ReauthInterval keyword is ignored when IKE version 1 is being used between IKE peers.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

Tip: When the remote IKE version 2 peer initiates a reauthentication at the same time as the local IKE version 2 peer, it is called a simultaneous reauthentication. Simultaneous reauthentication often results in redundant SAs. To reduce the probability of a simultaneous reauthentication, IKED shifts the timing of reauthentication by a small, random length of time. To further reduce the probability of a simultaneous reauthentication, use a higher ReauthInterval value or configure only one peer to initiate reauthentication.

FilterByIdentity
Indicates whether the peer's IKE identity is used for IP filtering purposes. IpFilterRule objects support the specification of a RemoteIdentity parameter. When this value is Yes, all IP tunnels negotiated with this peer use the RemoteIdentity parameter in addition to the traffic specification to locate the appropriate dynamic anchor IpFilterRule. When this value is No, all IP tunnels negotiated with this peer do not use the RemoteIdentity parameter to locate the appropriate dynamic anchor.

Restrictions:

- Because the RemoteIdentity parameter is supported only in combination with remote activation, FilterByIdentity Yes can be used only in combination with HowToInitiate DoNot.
- The peer is restricted to negotiating data protection only for its security endpoint address. RemoteIdentity support is intended for mobile users, who are not permitted to function as a security gateway.
- This parameter is valid only for V1R10 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

Guideline: When creating an IpFilterRule using a RemoteIdentity value, specify FilterByIdentity Yes on the KeyExchangeAction statement for the corresponding KeyExchangeRule statement. When creating an IPSec IpFilterRule without a RemoteIdentity value, specify FilterByIdentity No on the KeyExchangeAction statement for the corresponding KeyExchangeRule statement.

ConstrainSource
Indicates a source IP address constraint specification. Dynamic tunnel negotiations that take place under this KeyExchangeAction statement are constrained to include source data addresses that are in the range of this specification.

ipaddress
A single IP address constraining the source data address for all dynamic tunnel negotiations under this KeyExchangeAction statement.

ipaddress/prefixLength
A prefix address specification indicating the applicable source data addresses that can be included in dynamic tunnel negotiations under this KeyExchangeAction statement. The *prefixLength* value is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. A dynamic tunnel negotiation matches this condition if its source data address specification is entirely contained in the range defined by the unmasked bits for this prefix specification.

ipaddress-ipaddress
The range of IP addresses that are applicable source data addresses that can be included in dynamic tunnel negotiations under this KeyExchangeAction statement.

All
Indicates that dynamic tunnel negotiations under this KeyExchangeAction statement can include any IPv4 source data address specification. All and All4 are interchangeable values.

All4
Indicates that dynamic tunnel negotiations under this KeyExchangeAction statement can include any IPv4 source data address specification.

All6 Indicates that dynamic tunnel negotiations under this KeyExchangeAction statement can include any IPv6 source data address specification.

Restriction: This parameter, and the ConstrainSourceRef, ConstrainSourceSetRef, and ConstrainSourceGroupRef parameters are valid only for V1R10 and later releases. See“General syntax rules for Policy Agent” on page 933 for details.

ConstrainSourceRef
The name of a globally defined IpAddr statement that you should use to specify the source data address constraint.

ConstrainSourceSetRef
The name of a globally defined IpAddrSet statement that you should use to specify the source data address prefix or range constraint.

ConstrainSourceGroupRef
The name of a globally defined IpAddrGroup statement that you can use to specify the source data address constraint.

ConstrainDest
Indicates a destination IP address constraint specification. Dynamic tunnel negotiations that take place under this KeyExchangeAction statement are constrained to include only destination data addresses that are in the range of this specification.

ipaddress
A single IP address that constrains the destination data address for all dynamic tunnel negotiations under this KeyExchangeAction statement.

ipaddress/prefixLength
A prefix address specification indicating the applicable destination data addresses that can be included in dynamic tunnel negotiations under this KeyExchangeAction statement. The *prefixLength* value is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. A dynamic tunnel negotiation matches this condition if its destination data address specification is entirely contained within the range defined by the unmasked bits for this prefix specification.

ipaddress-ipaddress
The range of IP addresses that are applicable destination data addresses that can be included in dynamic tunnel negotiations under this KeyExchangeAction statement.

All Indicates that dynamic tunnel negotiations under this KeyExchangeAction statement can include any IPv4 destination data address specification. All and All4 are interchangeable values.

All4 Indicates that dynamic tunnel negotiations under this KeyExchangeAction statement can include any IPv4 destination data address specification.

All6 Indicates that dynamic tunnel negotiations under this KeyExchangeAction statement can include any IPv6 destination data address specification.

Restriction: This parameter, and the ConstrainDestRef, ConstrainDestSetRef, and ConstrainDestGroupRef parameters are valid only for V1R10 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

ConstrainDestRef *name*
The name of a globally defined IpAddr statement to be used for the destination data address constraint.

ConstrainDestSetRef *name*
The name of a globally defined IpAddrSet statement to be used for the destination data address prefix or range constraint.

ConstrainDestGroupRef *name*
The name of a globally defined IpAddrGroup statement to be used for the destination data address constraint.

BypassIpValidation
Indicates whether a check should be made to verify that the remote peer's identity matches the peer's remote IP address. A value of Yes indicates the check should be bypassed. A value of No indicates the check should be enforced. If this parameter is not specified, the BypassIpValidation setting from the KeyExchangePolicy statement is used as the default.

Restriction: The BypassIpValidation keyword is ignored when identity of the peer is not an IPv4 or IPv6 address.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Tip: If the remote security endpoint is expected to be behind a NAT, specify a value of Yes.

CertificateURLLookupPreference
Indicates the hash and URL encoding preference of certificate payloads. If this parameter is not specified, the CertificateURLLookupPreference setting from the KeyExchangePolicy statement is used as the default.

Allow IKED provides the remote security endpoint with an indication that it prefers to receive certificate payloads encoded in a hash and URL format. IKED processes certificate payloads encoded using a hash and URL format when they are received. IKED attempts to send certificate payloads using a hash and URL format encoding when the remote security endpoint indicates a preference to receive certificate payloads encoded in a hash and URL format.

Tolerate
IKED does not provide the remote security endpoint with an indication that it prefers to receive certificate payloads encoded in a hash and URL format. IKED processes certificate payloads encoded using a hash and URL format when they are received. IKED attempts to send certificate payloads using a hash and URL format encoding when the remote security endpoint indicates a preference to receive certificate payloads encoded in a hash and URL format.

Disallow
IKED does not provide the remote security endpoint with an indication that it prefers to receive certificate payloads encoded in a hash and URL format. IKED ignores certificate payloads encoded using a hash and URL format when they are received. IKED does not send certificate payloads using a hash and URL format.

Restriction: This keyword is ignored when IKE version 1 IKE SAs are negotiated since IKE version 1 does not support hash and URL encoding of certificate data.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

`RevocationChecking`

Indicates the level of revocation checking to be performed on a remote security endpoint's certificate and its corresponding certificate authority certificates.

None	No revocation checking is performed.
Loose	Revocation information is checked if available.
Strict	Revocation information must be available for all certificates and is checked for all certificates

If this parameter is not specified, the RevocationChecking setting from the KeyExchangePolicy statement is used as the default. .

Rules:

- Revocation checking is only applicable to digital signature authentication methods.
- When the mode is Loose and revocation information for a certificate is unavailable, then that certificate is considered valid.
- When the mode is Strict and revocation information for a certificate is unavailable, then that certificate is considered invalid.
- When the mode is Strict or Loose and a source of revocation information checked indicates that a certificate is revoked then the certificate is considered invalid.
- If a CRL can be obtained using the CRLDistributionPoints extension and a certificate bundle file, the CRL obtained from the CRLDistributionPoints extension is used and the CRL in the certificate bundle or certificate payload is ignored.
- When IKED is configured to use the native IKE daemon certificate service the RevocationChecking parameter is ignored.

Rule: Certificate revocation lists (CRL) received in a certificate payload are ignored.

Restrictions:

- This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.
- Certificate revocation lists (CRL) are the only source of revocation information consulted. The CRL must be identified in the CRLDistributionPoints extension of the certificate being checked or contained in a certificate bundle file identified by the remote security endpoint.
- When the CRLDistributionPoints extension is used to retrieve a CRL at least one distribution point must contain an HTTP URL.
- IKED will only consult a CRL that contain entries for all revocation reasons.
- The native IKE daemon certificate service does not consult certificate revocation information when authenticating a digital signature. If certificate revocation information is consulted then IKED must be configured as a network security client.

KeyExchangeGroup statement

Use the KeyExchangeGroup statement to define a key exchange group. A KeyExchangeGroup statement identifies a set of KeyExchangeRule statements that make up the key exchange group. A globally defined KeyExchangeGroup statement can be referenced by a KeyExchangePolicy statement.

Syntax

```
►►──KeyExchangeGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬──────────────────────────────────────┤
   │ ┌────────────────────────────┐ │
   │ ▼                            │ │
   ├───┬─KeyExchangeRuleRef name─┬┴─┤
   │   └─KeyExchangeRule─────────┘  │
   └─}──────────────────────────────┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this KeyExchangeGroup statement.

KeyExchangeRuleRef
: The name of a globally defined KeyExchangeRule statement to be included in the group.

KeyExchangeRule
: An inline specification of a KeyExchangeRule statement to be included in this group.

KeyExchangeOffer statement

Use the KeyExchangeOffer statement to define a key exchange offer for a dynamic VPN. A key exchange offer indicates one acceptable way to protect a key exchange for a dynamic VPN. A key exchange offer can be referenced by a KeyExchangeAction statement.

Syntax

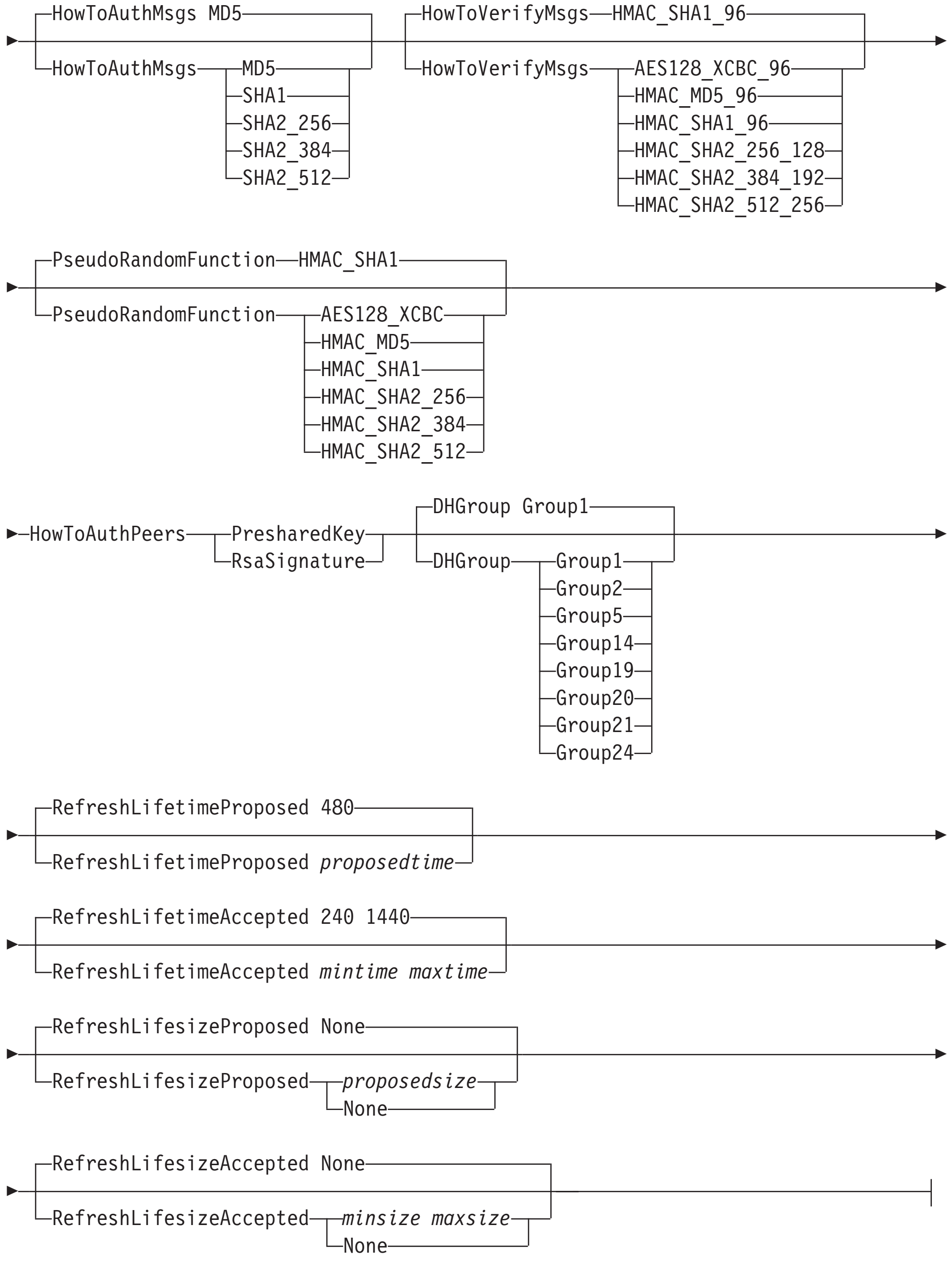
```
►►──KeyExchangeOffer──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                      └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬──────────────────────────────────────┤
   ├─┤ KeyExchangeOffer Parameters ├┤
   └─}──────────────────────────────┘
```

KeyExchangeOffer Parameters:

```
   ┌─HowToEncrypt──DES──────────────────────┐
├──┴─HowToEncrypt──┬─DES───────────────────┬┴──────────────────────────►
                   ├─3DES──────────────────┤
                   ├─AES───────────────────┤
                   └─AES_CBC KeyLength keylen┘
```



Parameters

name

A string 1 - 32 characters in length specifying the name of this KeyExchangeOffer statement.

Rule: If this KeyExchangeOffer statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline KeyExchangeOffer statement, a nonpersistent system name is created.

HowToEncrypt

The desired encryption policy for protecting key exchanges. The default is **DES**.

DES Use DES encryption, which uses a 56–bit key and a 64–bit initialization vector.

Restriction: DES is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

3DES Triple DES runs the DES encryption algorithm three times and uses 192-bits, including 24 parity bits.

Rule: If 3DES is specified but is not supported by the system, then the Policy Agent fails the policy.

AES Deprecated and treated as a synonym for AES_CBC KeyLength 128.

Rule: If AES is specified but AES encryption in CBC mode is not supported by this TCP/IP stack, Policy Agent fails the policy.

AES_CBC
The AES algorithm is used in Cipher Block Chaining (CBC) mode.

Rules:

- The key length is measured in bits, and a *keylen* of either 128 or 256 must be specified.
- If AES_CBC is specified but AES encryption is not supported by this TCP/IP stack, Policy Agent fails the policy.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

HowToAuthMsgs
The desired hash algorithm for authenticating IKE version 1 key exchange messages. The default is **MD5**.

MD5 Use the HMAC MD5 algorithm.

Restriction: MD5 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

SHA1 Use the HMAC_SHA1 algorithm.

SHA2_256
Use the HMAC_SHA2_256_128 algorithm.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

SHA2_384
Use the HMAC_SHA2_384_192 algorithm.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

SHA2_512
Use the HMAC_SHA2_512_256 algorithm.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Restriction: The HowToAuthMsgs parameter is ignored for IKE version 2 SAs.

HowToVerifyMsgs
The desired authentication algorithm for verifying message integrity of IKE version 2 key exchange messages. The default is HMAC_SHA1_96.

AES128_XCBC_96
Use the AES128_XCBC algorithm to encode authentication data, with 128-bit keys and hash truncation to 96 bits.

Restriction: AES128_XCBC_96 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

HMAC_MD5_96
Use the HMAC_MD5_96 algorithm.

Restriction: HMAC_MD5_96 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

HMAC_SHA1_96
Use the HMAC_SHA1_96 algorithm.

HMAC_SHA2_256_128
Use the HMAC_SHA2_256 algorithm to encode authentication data, with 256-bit keys and hash truncation to 128 bits.

HMAC_SHA2_384_192
Use the HMAC_SHA2_384 algorithm to encode authentication data, with 384-bit keys and hash truncation to 192 bits.

HMAC_SHA2_512_256
Use the HMAC_SHA2_512 algorithm to encode authentication data, with 512-bit keys and hash truncation to 256 bits.

Restrictions:
- The HowToVerifyMsgs parameter is ignored for IKE version 1 SAs.
- This HowToVerifyMsgs parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

`PseudoRandomFunction`
Indicates which pseudo-random function (PRF) to use when generating keying material for IKE version 2 SAs. The default is HMAC_SHA1.

AES128_XCBC
Use the AES128_XCBC algorithm.

Restriction: AES128_XCBC is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

HMAC_MD5
Use the HMAC_MD5 algorithm.

Restriction: HMAC_MD5 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

HMAC_SHA1
Use the HMAC_SHA1 algorithm.

HMAC_SHA2_256
Use the HMAC_SHA2_256 algorithm

HMAC_SHA2_384
Use the HMAC_SHA2_384 algorithm.

HMAC_SHA2_512
Use the HMAC_SHA2_512 algorithm.

Restrictions:
- The PseudoRandomFunction parameter is ignored for IKE version 1 SAs. IKE version 1 always uses the algorithm specified on HowToAuthMsgs to

determine its pseudo-random function. For example, if the HowToAuthMsgs value is MD5, then HMAC_MD5 is used.

- This PseudoRandomFunction parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

HowToAuthPeers

Specifies the method for authenticating peers during IKE version 1 phase 1 negotiation.

PresharedKey

Use a pre-shared key to authenticate the peer.

RsaSignature

Use an RSA signature to authenticate the peer.

Restriction: The HowToAuthPeers parameter is ignored for IKE version 2 SAs.

DHGroup

Specifies the Diffie-Hellman group used during the phase 1 key exchange. The default is **Group1**.

Group1

Modular exponentiation group with a 768-bit modulus.

Restriction: Group1 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group2

Modular exponentiation group with a 1024-bit modulus.

Restriction: Group2 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group5

Modular exponentiation group with a 1536-bit modulus.

Restriction: Group5 is not accepted when the TCP/IP stack is configured for FIPS 140 mode on the IpFilterPolicy statement.

Group14

Modular exponentiation group with a 2048-bit modulus.

Group19

Random 256-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group20

Random 384-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group21

Random 521-bit elliptic curve group.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Group24

Modular exponentiation group with a 2048-bit modulus and 256-bit prime order subgroup.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Guideline: If you are using encryption or authentication algorithms with a 128-bit key, use Diffie-Hellman groups 5,14,19,20, or 24. If you are using encryption or authentication algorithms with a key length of 256 bits or greater, use Diffie-Hellman group 21.

Tip: When negotiating a new phase 1 SA and when the negotiation mode is IKE version 1 aggressive mode, only the first offer and its DH group are proposed to the peer. If the negotiation mode is IKE version 1 main mode, all offers and DH groups are proposed to the peer, who will select a particular offer and group. If the negotiation uses IKE version 2, then all offers and DH groups will be proposed, but only one DH group will be calculated in the proposal. The peer is free to either accept the DH group value used or choose a different value from one of the other offers. In that case, the IKE daemon starts the exchange again using the chosen group.

`RefreshLifetimeProposed`

The security association lifetime in minutes. This value is proposed when acting as the IKE version 1 initiator of a key exchange negotiation. For IKE version 2, this value determines the refresh lifetime. The default is 480.

proposedtime

The lifetime proposed (for IKE version 1) or used (for IKE version 2) for the phase 1 tunnel. Valid values are in the range 1 - 9 999. The proposed lifetime value should be within the range specified by RefreshLifetimeAccepted.

Tip: When negotiating an IKE version 2 SA, the IKE daemon uses the RefreshLifetimeProposed value in the first matching offer for the SA lifetime. Unlike IKE version 1, SA lifetimes are not negotiated under IKE version 2.

`RefreshLifetimeAccepted`

A range of acceptable security association lifetimes in minutes. This range is accepted when acting as the responder of an IKE version 1 key exchange negotiation. The default is 240 1440.

mintime

The minimum lifetime that can be accepted.

maxtime

The maximum lifetime that can be accepted. This value must be ≥ to the *mintime* value.

Valid values for each option are in the range 1 - 9 999.

Restriction: The RefreshLifetimeAccepted parameter is ignored for IKE version 2 SAs.

`RefreshLifesizeProposed`

The security association lifesize in Kilobytes. If a *proposedsize* value is specified, then this value is proposed when acting as the IKE version 1 initiator of a key exchange negotiation. For IKE version 2, this value determines the refresh lifesize. If **None** is specified, then no lifesize is proposed for IKE version 1 or used for IKE version 2. The default is **None**.

proposedsize

The proposed lifesize for the negotiation. Valid values are in the range 1 - 4 194 300. The proposed lifetime value should be within the range specified by RefreshLifesizeAccepted value, if that parameter is not specified as **None**.

None No lifesize should be proposed for IKE version 1 or used for IKE version 2. If the RefreshLifesizeProposed parameter is specified as **None**, then RefreshLifesizeAccepted value should also be specified as **None**.

Tip: When negotiating an IKE version 2 SA, the IKE daemon uses the RefreshLifesizeProposed value in the first matching offer for the SA lifesize. Unlike IKE version 1, SA lifesizes are not negotiated under IKE version 2.

`RefreshLifesizeAccepted`

The security association lifesize in Kbytes. If *minsize* and *maxsize* values are specified, this range is accepted when acting as the responder of key exchange negotiation. If **None** is specified, no lifesize is accepted when acting as the responder of a key exchange negotiation. The default is **None**.

minsize

The minimum lifesize that can be accepted.

maxsize

The maximum lifesize that can be accepted. This value must be ≥ to the *minsize* value.

None No lifesize is accepted. If this parameter is specified as **None**, then RefreshLifesizeProposed should also be specified as **None**.

Valid values for the *minsize* and *maxsize* options are in the range 1 - 4 194 300.

Restriction: The RefreshLifesizeAccepted parameter is ignored for IKE version 2 SAs.

KeyExchangePolicy statement

Use the KeyExchangePolicy statement to define a key exchange policy. The Key exchange policy is consulted when creating a phase 1 security association for a dynamic VPN. The KeyExchangePolicy statement can contain a combination of references to global KeyExchangeGroup statements, references to global KeyExchangeRule statements, and inline KeyExchangeRule statements.

When acting as the responder of an IKE version 1 main mode or an IKE version 2 phase 1 exchange, the IKE daemon continues negotiation without knowing the identity of the remote endpoint, as long as the suggested algorithms are supported by the IKE daemon.

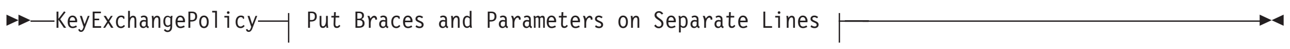
Any SA agreed to before the identity of both parties are known is verified when the identity of both parties become known. If the SA is not consistent with the defined key exchange policy, the phase 1 negotiation fails. If the SA is consistent with the defined key exchange policy, the phase 1 negotiation continues.

The KeyExchangePolicy statement can appear in the common IPSec policy file, a stack-specific IPSec policy file, or both. If it appears in both, Policy Agent only uses the statement contained in the stack-specific IPSec policy file. It should appear at most only once in each file. If it appears multiple times in a file, the last one encountered is used.

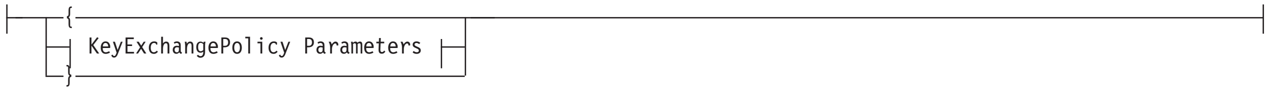
Requirement: The KeyExchangePolicy statement is required to define key exchange policies to the Policy Agent.

Result: If the KeyExchangePolicy statement is deleted, then all KeyExchange policies are deleted from the IKE daemon for the corresponding stack.

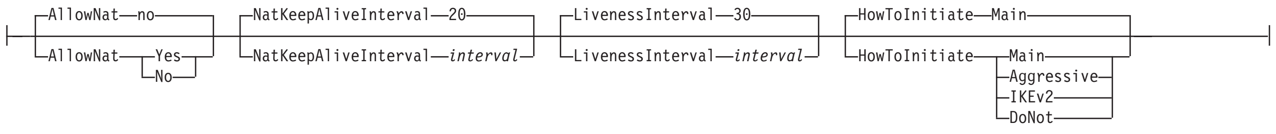
Syntax

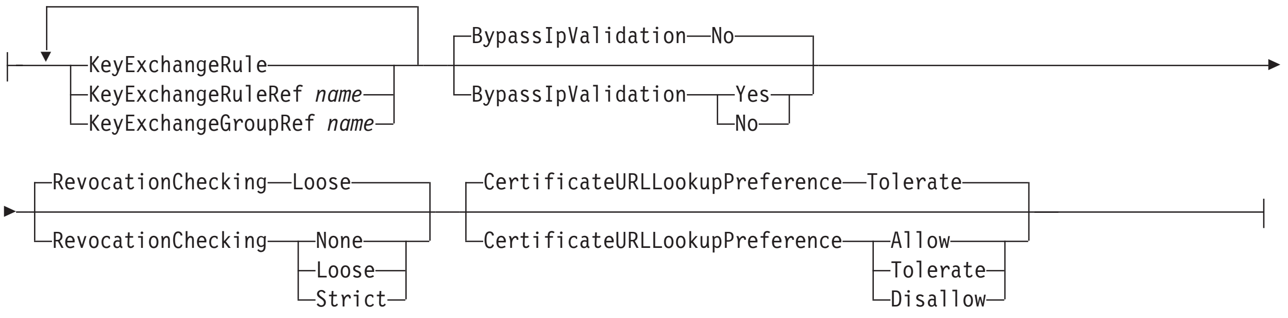


Put Braces and Parameters on Separate Lines:



KeyExchangePolicy Parameters:





Parameters

`AllowNat`

Indicates whether negotiations that use NAT traversal techniques are allowed when the AllowNat parameter is omitted from a **KeyExchangeAction** statement. The value Yes indicates that negotiations that use NAT traversal techniques are allowed when the AllowNat parameter is omitted from a **KeyExchangeAction** statement. The value No indicates that negotiations that use NAT traversal techniques are not allowed when the AllowNat parameter is omitted from a **KeyExchangeAction** statement. The default value is No.

Rule: If you change the AllowNat setting, the change is effective immediately for any new phase 1 security associations (SAs) that are negotiated. For existing phase 1 SAs, the change takes effect when the phase 1 SA is refreshed.

Tip: Setting the AllowNat to No prevents the IKE daemon from sending NAT payloads or processing received NAT payloads as part of the tunnel negotiation. In some cases. tunnels traversing one or more NATs can still be activated even when AllowNat is set to No. However, such tunnels are normally unusable because of the known incompatibilities between IPsec and NAT documented in RFC 3715.

`NatKeepAliveInterval`

When the IKE server is behind a NAT device it might need to send NAT keepalive messages to remote security endpoints. These keepalive messages must be sent to a remote security endpoint when all of the following conditions are true:

- IKE is behind a NAT device that dynamically maps IKE's IP address to a public IP address.
- A phase 1 SA exists with that remote security endpoint.
- No other packets have been sent to that remote security endpoint within a configured inactivity interval.

The purpose of a NAT keepalive message is to prevent a NAT device from expiring dynamic NAT mappings. NAT keep alive messages are not required when no NAT device is in front of the IKE server or when the NAT device in front of the IKE server statically maps the IKE servers IP address to a public IP address.

interval

The configured inactivity interval in seconds. Valid values are 0 or within the range 20-999. A value of 0 indicates that NAT keepalive messages should never be sent. A value in the range 20 - 999 indicates the number of seconds of inactivity that triggers the sending of a NAT keepalive message. The default is 20 seconds.

Rule: A change to the NatKeepAliveInterval interval is effective immediately for any new timer started. For existing timers a change takes effect when the timer expires. It is rescheduled with the new *interval* value.

Tip: The following should be considered in defining the interval value for the NatKeepAliveInterval parameter:

- The KeepAlive timer runs only if there is a NAT device in front of z/OS.
- If a static NAT device is in front of z/OS, then a value of 0 should be defined for the interval.
- If a dynamic NAT device is in front of z/OS, then a value less than the mapping expiration of the NAT device should be defined.

`LivenessInterval`

When IKE negotiates a security association using IKE version 2 (IKEv2), IKE can perform periodic liveness checks as prescribed in RFC 5996 to test whether the peer remains active. When traffic was sent but not received on an IKE SA or any of its associated dynamic tunnels, IKE initiates a liveness check after the liveness interval period is exceeded. IKE does this by sending the peer a request to which the peer is expected to respond. If, after the normal series of IKE retransmissions, the peer has not responded, the IKE SA and associated dynamic tunnels are considered non-responsive and are deleted. See the IkeInitWait parameter and the IkeRetries parameter for more information.

The purpose of the liveness check is to verify whether an IKEv2 peer is still active. This allows IKE to detect cases such as when a peer has rebooted or when dynamic tunnels are non-responsive. This allows IKE to re-establish communications with the peer in a timely manner.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

interval

The configured liveness interval in seconds. Valid values are 0 - 20999. A value of 0 indicates that liveness checks should never be performed.

A value in the range 1 - 20999 indicates the number of seconds of inactivity that triggers the sending of liveness check request. The default is 30 seconds.

Rule: A change to the LivenessInterval is effective immediately for any new timer started. For existing timers a change takes effect when the timer expires. It is rescheduled with the new *interval* value.

Results:

1. LivenessInterval is used only for IKE version 2 security associations. This value is ignored for IKE version 1 security associations.
2. Liveness checks are performed only when data has been sent over the IKE SA or dynamic tunnels but not received. If no data is being either sent or received, then no liveness checks are performed.

`HowToInitiate`
The negotiation mode to use when initiating a KeyExchangeAction statement does not contain a HowToInitiate parameter. The default is Main.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Main Indicates that IKE version 1 with identity protection is used when key negotiations are initiated by this system.

Aggressive
Indicates that IKE version 1 without identity protection is used when key negotiations are initiated by this system.

IKEv2 Indicates that IKE version 2 is used when key negotiations are initiated by this system.

DoNot
Indicates that the local system cannot initiate a key exchange negotiation.

`KeyExchangeRule`
An inline specification of a KeyExchangeRule statement to be included in the policy.

`KeyExchangeRuleRef`
The name of a globally defined KeyExchangeRule statement to be included in the policy.

`KeyExchangeGroupRef`
The name of a globally defined KeyExchangeGroup statement to be included in the policy.

`BypassIpValidation`
Indicates whether a check should be made to verify that the remote peer's identity matches the peer's remote IP address. A value of Yes indicates the check should be bypassed. A value of No indicates the check should be enforced. The default is No,

Restriction: The BypassIpValidation keyword is ignored when the identity of the peer is not an IPv4 or IPv6 address.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Tip: If the remote security endpoint is expected to be behind a NAT, specify a value of Yes.

CertificateURLLookupPreference
Indicates hash and URL encoding preference in certificate payloads and certificate request payloads sent and received.

Allow IKED provides the remote security endpoint with an indication that it prefers to receive certificate payloads encoded in a hash and URL format. IKED processes certificate payloads encoded using a hash and URL format when they are received. IKED attempts to send certificate payloads using a hash and URL format encoding when the remote security endpoint indicates a preference to receive certificate payloads encoded in a hash and URL format.

Tolerate
IKED does not provide the remote security endpoint with an indication that it prefers to receive certificate payloads encoded in a hash and URL format. IKED processes certificate payloads encoded using a hash and URL format when they are received. IKED attempts to send certificate payloads using a hash and URL format encoding when the remote security endpoint indicates a preference to receive certificate payloads encoded in a hash and URL format. This is the default value.

Disallow
IKED does not provide the remote security endpoint with an indication that it prefers to receive certificate payloads encoded in a hash and URL format. IKED ignores certificate payloads encoded using a hash and URL format when they are received. IKED does not send certificate payloads using a hash and URL format.

Restriction: This keyword is ignored when IKE version 1 IKE SAs are negotiated since IKE version 1 does not support hash and URL encoding of certificate data.

Restriction: This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

RevocationChecking
Indicates the level of revocation checking to be performed on a remote security endpoint's certificate and its corresponding certificate authority certificates. The default is Loose.

None No revocation checking is performed.

Loose Revocation information is checked if available.

Strict Revocation information must be available for all certificates and is checked for all certificates

Rules:

- Revocation checking is only applicable to digital signature authentication methods.
- When the mode is Loose and revocation information for a certificate is unavailable, then that certificate is considered valid.
- When the mode is Strict and revocation information for a certificate is unavailable, then that certificate is considered invalid.
- When the mode is Strict or Loose and a source of revocation information checked indicates that a certificate is revoked then the certificate is considered invalid.

- If a CRL can be obtained using the CRLDistributionPoints extension and a certificate bundle file, the CRL obtained from the CRLDistributionPoints extension is used and the CRL in the certificate bundle or certificate payload is ignored.
- When IKED is configured to use the native IKE daemon certificate service the RevocationChecking parameter is ignored.

Rule: A CRL received in a certificate payload is ignored.

Restrictions:

- This parameter is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.
- Certificate revocation lists (CRL) are the only source of revocation information consulted. The CRL must be identified in the CRLDistributionPoints extension of the certificate being checked or contained in a certificate bundle file identified by the remote security endpoint.
- When the CRLDistributionPoints extension is used to retrieve a CRL at least one distribution point must contain an HTTP URL.
- IKED will only consult a CRL that contain entries for all revocation reasons.
- The native IKE daemon certificate service does not consult certificate revocation information when authenticating a digital signature. If certificate revocation information is consulted then IKED must be configured as a network security client.

KeyExchangeRule statement

An IKE SA establishment might be initiated from the local system or from a remote system, and it involves several message exchanges. Depending on the initiator/responder state, and the message sequence, the IKE daemon locates a KeyExchangeRule statement to govern the policy that is used during the negotiation. The following base values are used in some combination to locate a KeyExchangeRule statement:

- Local IP address
- Local ID value
- Remote IP address
- Remote ID value

Depending on the message sequence, one or more of the base values might not be available, but the KeyExchangeRule statement lookup returns the best rule match available.

Syntax

```
►►──KeyExchangeRule──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├───┬─{──────────────────────────────┬────────────────────────────────────────────┤
    ├─┤ KeyExchangeRule Parameters ├─┤
    └─}──────────────────────────────┘
```

KeyExchangeRule Parameters:

```
├──┬─LocalSecurityEndpoint──────────────┬──┬─RemoteSecurityEndpoint──────────────┬──────────►
   └─LocalSecurityEndpointRef name─┘     └─RemoteSecurityEndpointRef name─┘

►─KeyExchangeActionRef name──┬─────────────────────────────────┬────────────────────┤
                             └─SharedKey──┬─Ascii string───┬─┘
                                          ├─Ebcdic string──┤
                                          └─Hex hexstring──┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this KeyExchangeRule statement. The name cannot start with a dash (-) or contain any commas (,).

LocalSecurityEndpoint
An inline specification of a LocalSecurityEndpoint statement.

LocalSecurityEndpointRef
The name of a globally defined LocalSecurityEndpoint statement.

RemoteSecurityEndpoint
An inline specification of a RemoteSecurityEndpoint statement.

RemoteSecurityEndpointRef
The name of a globally defined RemoteSecurityEndpoint statement.

KeyExchangeActionRef
The name of a globally defined KeyExchangeAction statement.

SharedKey
The shared key to use with the LocalSecurityEndpoint statement and RemoteSecurityEndpoint statement pair when using a pre-shared key for authentication. The maximum length for an ASCII or EBCDIC string is 900 characters. The maximum length for a hexadecimal string is 450 bytes. The hexstring must begin with a 0x.

Examples:

SharedKey Ascii SharedKeyValue
The value is treated as an ASCII string. This specification is valuable if the sharedkey has been defined to the other endpoint as an ASCII string.

SharedKey Ebcdic SharedKeyValue
The value is treated as an EBCDIC string.

SharedKey Hex 0xC1C2C3F1F2F3
The value is treated as a hexadecimal string.

The ASCII or EBCDIC SharedKey value can be defined as a quoted string or a single value.

Rules:

- A quoted string must start and end with a double-quote (").
- A quoted string allows the SharedKey value to have embedded blanks for the attribute.
- If SharedKey value is not a quoted string then it as treated as a single value.

Results:

- Leading blanks and trailing blanks within the quoted string are removed.
- Within a quoted string, comment indicators, embedded blanks, and additional quotes are treated as part of the value for this attribute.

Restriction: When the value contains embedded blanks, you must specify the entire parameter value within the first 1-536 characters of the configuration file line.

Example SharedKey values:

```
SharedKey ASCII ASC # comment" value used: ASC
SharedKey EBCDIC EBC comment value used: EBC
SharedKey ASCII "ASC 98Z" value used: ASC 98Z
SharedKey EBCDIC EBC 98Z" value used: EBC
SharedKey ASCII "AsC 98Z value used: "AsC
SharedKey EBCDIC "Ebc " " Ebc" value used: Ebc " " Ebc
SharedKey ASCII "Asc Asc" " value used: Asc Asc
```

Tip: When negotiating using IKE version 1, the authentication method used in both directions is determined by the HowToAuthPeers parameter on the KeyExchangeOffer statement. When negotiating using IKE version 2, the IKE peers may choose different authentication methods. If you are negotiating using IKE version 2, the HowToAuthPeers parameter is ignored, and instead the HowToAuthMe parameter on the KeyExchangeAction statement determines the authentication method that the IKED uses for its local identity. When negotiating using IKE version 2, the peer will choose its own authentication method. If either the IKED or the remote IKE peer uses a pre-shared key for authentication, that key is to be configured using the SharedKey statement.

Rule: The z/OS IKE daemon requires that both security endpoints use the same pre-shared key value to authenticate itself to the remote security endpoint and to authenticate the remote security endpoint's identity.

Rule: If the z/OS IKE daemon is enabled for FIPS 140 mode, the pre-shared key's length must be at least half the length of the key size of the chosen KeyExchangeOffer's HowToAuthMsgs (IKEv1) or PseudoRandomFunction (IKEv2) algorithm.

Rule: All Location addresses in the LocalSecurityEndpoint and RemoteSecurityEndpoint parameters for this rule, as well as all IP addresses in the ConstrainSource and ConstrainDest specification for this rule's action, must be in the same address family (IPv4 or IPv6).

LocalDynVpnGroup statement

Use the LocalDynVpnGroup statement to define a local dynamic VPN group. A LocalDynVpnGroup statement identifies a set of LocalDynVpnRule statements that make up the local dynamic VPN group. A globally defined LocalDynVpnGroup statement can be referenced by a LocalDynVpnPolicy statement.

Syntax

```
►►──LocalDynVpnGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬──────────────────────────────────────────┤
   │   ┌──────────────────────────────┐  │
   │   ▼                              │  │
   ├────┬─LocalDynVpnRuleRef name─┬───┴──┤
   │    └─LocalDynVpnRule─────────┘      │
   └─}───────────────────────────────────┘
```

Parameters

name
: A string 1 - 32 characters in length the name of this LocalDynVpnGroup.

LocalDynVpnRuleRef
: The name of a globally defined LocalDynVpnRule statement to be included in the group.

LocalDynVpnRule
: An inline specification of a LocalDynVpnRule statement to be included in this group.

LocalDynVpnPolicy statement

Use the LocalDynVpnPolicy statement to identify a set of LocalDynVpnRule and LocalDynVpnGroup statements that make up the local dynamic VPN policy.

The LocalDynVpn Policy statement can appear in the common IPSec policy file, a stack-specific IPSec policy file, or both. If it appears in both, Policy Agent only uses the statement contained in the stack-specific IPSec policy file. It should appear at most only once in each file. If it appears multiple times in a file, the last one encountered is used.

Requirement: The LocalDynVpnPolicy statement is required in order to define local dynamic VPN policies to the Policy Agent.

Syntax

```
►►──LocalDynVpnPolicy──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{────────────────────────────────┬────────────────────────────────────────┤
   ├─┤ LocalDynVpnPolicy Parameters ├─┤
   └─}────────────────────────────────┘
```

LocalDynVpnPolicy Parameters:

```
      ┌──────────────────────────────┐
      ▼                              │
├────┬─LocalDynVpnGroupRef name─┬───┴────────────────────────────────────────────┤
     ├─LocalDynVpnRuleRef name──┤
     └─LocalDynVpnRule──────────┘
```

Parameters

`LocalDynVpnRule`
An inline specification of a LocalDynVpnRule statement to be included in the policy.

`LocalDynVpnRuleRef`
The name of a globally defined LocalDynVpnRule statement to be included in the policy.

`LocalDynVpnGroupRef`
The name of a globally defined LocalDynVpnGroup statement to be included in the policy.

Result: If the LocalDynVpnPolicy statement is deleted, then all LocalDynVpn policies are deleted from the IKE daemon for the corresponding stack.

LocalDynVpnRule statement

Use the LocalDynVpnRule statement to specify the parameters that are used to negotiate a dynamic VPN that can be autoactivated or activated by an **`ipsec`** command. The parameters describe the traffic pattern of the data to be protected by the dynamic VPN.

Syntax

```
►►──LocalDynVpnRule──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{────────────────────────────┬─────────────────────────────────────┤
   ├─┤ LocalDynVpnRule Parameters ├─┤
   └─}────────────────────────────┘
```

LocalDynVpnRule Parameters:

```
├──┬─LocalIP──┬─ipaddress──────────────┬─┬──────────────────────────────►
   │          ├─ipaddress/prefixLength─┤ │
   │          ├─ipaddress-ipaddress────┤ │
   │          ├─All────────────────────┤ │
   │          ├─All4───────────────────┤ │
   │          └─All6───────────────────┘ │
   ├─LocalIpRef──name────────────────────┤
   └─LocalIpSetRef──name─────────────────┘

                                              ┌─AutoActivate No────────┐
►──┬─RemoteIP──┬─ipaddress──────────────┬─┬──┴─AutoActivate──┬─Yes─┬─┴──►
   │           ├─ipaddress/prefixLength─┤ │                  └─No──┘
   │           ├─ipaddress-ipaddress────┤ │
   │           ├─All────────────────────┤ │
   │           ├─All4───────────────────┤ │
   │           └─All6───────────────────┘ │
   ├─RemoteIpRef──name────────────────────┤
   └─RemoteIpSetRef──name─────────────────┘
   ┌─Protocol All──────────────────────────────────────┐
►──┼───────────────────────────────────────────────────┼──────────────────────────────┤
   └─Protocol────┬─┬─Tcp─┬─┤ PortSpecification ├─┬───┘
                 │ └─6───┘                        │
                 ├─┬─Udp─┬─┤ PortSpecification ├─┤
                 │ └─17──┘                        │
                 ├─Icmp───────────────────────────┤
                 ├─Icmpv6─────────────────────────┤
                 ├─Ip─────────────────────────────┤
                 ├─Ipip───────────────────────────┤
                 ├─Ah─────────────────────────────┤
                 ├─Esp────────────────────────────┤
                 ├─Ospf───────────────────────────┤
                 ├─Igmp───────────────────────────┤
                 ├─All────────────────────────────┤
                 └─n──────────────────────────────┘
```

PortSpecification:

```
   ┌─LocalDataPort 0──────┐   ┌─RemoteDataPort 0──────┐
├──┼──────────────────────┼───┼───────────────────────┼──────────────────────────┤
   └─LocalDataPort──port──┘   └─RemoteDataPort──port──┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this LocalDynVpnRule statement. The name cannot start with a dash (-) or contain any commas (,).

LocalIp
: Indicates the applicable local IP specification.

ipaddress
: A single IP address indicating the applicable local IP specification.

ipaddress/prefixLength
: A prefix address specification indicating the applicable local IP addresses. The *prefixLength* value is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses.

ipaddress-ipaddress
: A range of IP addresses indicating applicable local addresses.

All Any local IPv4 address is applicable. **All** and **All4** are interchangeable values.

All4 Any local IPv4 address is applicable.

All6 Any local IPv6 address is applicable.

LocalIpRef
: The name of a globally defined IpAddr statement to be used for the local IP address specification.

LocalIpSetRef
: The name of a globally defined IpAddrSet statement to be used for the local IP address specification.

RemoteIp

Indicates the applicable remote IP specification.

ipaddress
: A single IP address indicating the applicable remote IP specification.

ipaddress/prefixLength
: A prefix address specification indicating the applicable remote IP addresses. The *prefixLength* value is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP packet matches this condition if its remote address unmasked bits are identical to the defined unmasked bits.

ipaddress-ipaddress
: A range of IP addresses indicating applicable remote addresses.

All Any remote IPv4 address is applicable. **All** and **All4** are interchangeable values.

All4 Any remote IPv4 address is applicable.

All6 Any remote IPv6 address is applicable.

RemoteIpRef

The name of a globally defined IpAddr statement to be used for the remote IP address specification.

RemoteIpSetRef

The name of a globally defined IpAddrSet statement to be used for the remote IP address specification.

LocalDataPort

Indicates the applicable local port specification. Valid values for *n* are in the range 0 - 65 535. If 0 is specified for *n*, then the rule applies to any local port.

Restriction: This parameter can be specified only with Protocol TCP or Protocol UDP.

RemoteDataPort

Indicates the applicable remote port specification. Valid values for *n* are in the range 0 - 65 535. If 0 is specified for *n*, then the rule applies to any remote port.

Restriction: This parameter can be specified only with Protocol TCP or Protocol UDP.

Protocol

Indicates the protocol. If a numeric value is specified for *n*, it identifies an actual protocol number. The value for *n* must be less than 256.

If a value of **All** is specified, then the rule applies to any protocol.

The protocol name **Ip** maps to a value of 4, representing **ip in ip encapsulation**, for which the Internet Assigned Numbers Authority (IANA) has assigned the name **IP**.

The name **Ipip** maps to a value of 94, representing **ip within ip encapsulation**, for which IANA has assigned the name **IPIP**.

AutoActivate

If set to **Yes**, IKE attempts to activate a dynamic VPN tunnel to protect the IP traffic described by the LocalDynVpnRule statement. An activation is attempted when IKE connects to the corresponding stack and when an **ipsec -f reload** is issued for the stack associated with the LocalDynVpnRule statement.

Rules: In order for an autoactivation to occur the following conditions are required:

- A LocalDynVpnRule statement describing the traffic pattern to be protected must be defined and that statement must specify that the autoactivate option is in effect.
- The LocalDynVpnPolicy statement in effect for a stack must contain a reference to that LocalDynVpnRule statement. The reference can be a direct reference to the LocalDynVpnRule statement or a reference to a LocalDynVpnGroup statement containing the LocalDynVpnRule statement.
- A matching IP filter rule must exist and the action of that IP filter rule must be an IpDynVpnAction statement.
- A matching KeyExchangeRule value must exist.
- If IKE is configured for this stack as an NSS client for certificate services, the NSS server must be active and IKE must be connected to NSS before autoactivation is attempted.
- The remote security endpoint's IKE daemon must be active. If the IKE daemon initiates negotiation and the remote security endpoint does not respond, the IKE daemon uses its configuration parameters (IkeRetries and IkeInitWait on the IkeConfig statement) to control retry attempts. If the retries all fail, then the autoactivation fails.
- LocalIP and RemoteIP addresses must be in the same address family (IPv4 or IPv6).

Result: VPN policy is located by finding the first matching IP filter rule based on the LocalIP, RemoteIp, LocalDataPort, RemoteDataPort, and Protocol specifications.

LocalSecurityEndpoint statement

Use the LocalSecurityEndpoint statement to encapsulate a local security endpoint's IP address or host name and identity information.

Guideline: Do not use the same local identity on more than one TCP/IP stack or z/OS system image. IKED assumes that an identity is not used by any other stack, and this assumption can lead to a disruption of IPsec service for other stacks if identities are shared between stacks.

Syntax

```
►►──LocalSecurityEndpoint──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                           └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬──────────────────────────────────────┤
   ├──┤ LocalSecurityEndpoint Parameters ├─┤
   └─}──────────────────────────────────┘
```

LocalSecurityEndpoint Parameters:

```
├──Identity──┬─IpAddr authid──────────────┬──────────────────────────────────────────►
             ├─KeyID ──┬─Ascii authid───┤ │
             │         ├─Ebcdic authid──┤ │
             │         └─Hex authid─────┘ │
             ├─Fqdn authid────────────────┤
             ├─UserAtFqdn authid──────────┤
             └─X500dn authid──────────────┘

   ┌─Location──Any4────────────────────────┐
►──┼───────────────────────────────────────┼──────────────────────────────────────┤
   ├─Location──┬─ipaddress───────────────┬─┤
   │           ├─ipaddress/prefixLength──┤ │
   │           ├─ipaddress-ipaddress─────┤ │
   │           ├─Any─────────────────────┤ │
   │           ├─Any4────────────────────┤ │
   │           └─Any6────────────────────┘ │
   ├─LocationRef name──────────────────────┤
   ├─LocationSetRef name───────────────────┤
   └─LocationGroupRef name─────────────────┘
```

Parameters

name

A string 1 - 32 characters in length specifying the name of this LocalSecurityEndpoint statement.

Rule: If this LocalSecurityEndpoint statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline LocalSecurityEndpoint statement, a nonpersistent system name is created.

Identity

The identity of the local security endpoint. This identity cannot be wildcarded.

The following identity types and formats are supported:

IpAddr

Indicates that the *authid* value is an IP address, for example, 1.2.3.4 or 1::9.

KeyID

Indicates that the *authid* value is an opaque byte stream. This identity type is intended for use with pre-shared key authentication. The ID value can be specified as an ASCII string, an EBCDIC string, or a hexadecimal string. The maximum length for an ASCII or EBCDIC string is 900 characters. The maximum length for a hexadecimal string is 450 bytes. The hexstring must begin with a 0x.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Examples:

KeyID Ascii SharedKeyValue

The value is treated as an ASCII string. This specification is valuable if the key ID is defined to the other endpoint as an ASCII string.

KeyID Ebcdic SharedKeyValue
The value is treated as an EBCDIC string.

KeyID Hex 0xC1C2C3F1F2F3
The value is treated as a hexadecimal string.

The ASCII or EBCDIC KeyID value can be defined as a quoted string or a single value.

Rules:

- A quoted string must start and end with a double-quote (").
- A quoted string allows the KeyID value to have embedded blanks for the attribute.
- If KeyID value is not a quoted string then it as treated as a single value.

Results:

- Leading blanks and trailing blanks within the quoted string are removed.
- Within a quoted string, comment indicators, embedded blanks, and additional quotes are treated as part of the value for this attribute.

Restriction: When the value contains embedded blanks, you must specify the entire parameter value within the first 1 536 characters of the configuration file line.

Example KeyID values:

```
Identity KeyID Ascii   ASC # comment"  value used:  ASC
Identity KeyID EBCDIC  EBC comment     value used:  EBC
Identity KeyID ASCII   "ASC 98Z"       value used:  ASC 98Z
Identity KeyID EBCDIC  EBC 98Z"        value used:  EBC
Identity KeyID ASCII   "AsC 98Z        value used:  "AsC
Identity KeyID EBCDIC  "Ebc " " Ebc"   value used:  Ebc " " Ebc
Identity KeyID ASCII   "Asc Asc" "     value used:  Asc Asc"
```

Fqdn Indicates that a *authid* value is a fully qualified domain name or host name. For example: vnet.ibm.com. The maximum length accepted is 1024 characters. The Fqdn value cannot begin or end with a dot (.), or contain consecutive dots.

UserAtFqdn
Indicates that a *authid* value is a user at a fully qualified domain name or host name. The user name cannot contain a blank. For example: ibm@vnet.ibm.com. The maximum length accepted is 1024 characters. The UserAtFqdn value cannot begin or end with a dot (.), or contain consecutive dots.

X500dn
Indicates that the *authid* value is an X.500 distinguished name (DN). The DN must be specified in accordance with RFC 2253. A double-byte character is represented using the escaped UTF-8 encoding of the double-byte character in the Unicode character set. Attribute types can be specified using either attribute names or numeric object identifiers. Attribute values must represent string values. The maximum length accepted is 1024 characters.

Table 74 on page 1141 lists the DN attribute names that are recognized by the System SSL run time. An error is returned if the DN contains an unrecognized attribute name.

Table 74. DN attribute names

Attribute	Name
C	Country
CN	Common name
DC	Domain component
E	E-mail address
EMAIL	E-mail address (preferred)
EMAILADDRESS	E-mail address
L	Locality
O	Organization name
OU	Organizational unit name
PC	Postal code
S	State or province
SN	Surname
SP	State or province
ST	State or province (preferred)
STREET	Street
T	Title

The following code is an example of a DN using attribute names and string values:

```
CN=Ronald Hoffman,OU=Endicott,O=IBM,C=US
```

The following code is the same DN using object identifiers and encoded string values. The encoded string values represent the ASN.1 DER encoding of the string. The System SSL run time supports these ASN.1 string types:

```
PRINTABLE, VISIBLE, TELETEX, IA5, UTF8, BMP, and UCS.
2.5.4.3=#130E526F6E616C6420486F66666D616E,2.5.4.11=#1308456E6469636F7474,
2.5.4.10=#130349424D,2.5.4.6=#13025553
```

Individual characters can be represented using escape sequences. This is useful when the character cannot be represented in a single-byte character set. The hexadecimal value for the escape sequence is the UTF-8 encoding of the character in the Unicode character set. Table 75 shows some Unicode example letter descriptions.

Table 75. Unicode letter descriptions

Unicode letter description	10646 code	UTF-8	Quoted
LATIN CAPITAL LETTER L	U0000004C	0x4C	L
LATIN SMALL LETTER U	U00000075	0x75	u
LATIN SMALL LETTER C WITH CARON	U0000010D	0xC48D	\C4\8D
LATIN SMALL LETTER I	U00000069	0x69	i
LATIN SMALL LETTER C WITH ACUTE	U00000107	0xC487	\C4\87

Guidelines:

- The letters in the Quoted column in Table 75 on page 1141 can be used to encode a surname as follows:

  ```
  SN=Lu\C4\8Di\C4\87
  ```
- An X500dn type identity with a length of 256 characters is guaranteed to be accepted when it is DER-encoded. An X500dn type identity with a length approaching 1024 characters may not be accepted when it is DER-encoded.

An escape sequence can also be used for special characters that are part of the name and are not to be interpreted as delimiters. The following special characters must be represented as an escape sequence (prefixed with a backslash '\') when used as part of the name:

- A space or # character occurring at the beginning of the string
- A space character occurring at the end of the string
- One of the following characters , + " \ < > ;.

For example:

```
CN=L. Eagle,OU=Jones\, Dale and Mian,O=IBM,C=US
```

Rules:

- When an X500dn type identity is specified, the DN attributes must have the same order as those of the corresponding certificate subject name.
- When an X500dn type identity is parsed, it is converted to an ASN.1 stream and DER-encoded. The maximum length accepted for the DER-encoded name is 1024 bytes.
- You can use comment indicators and embedded blanks as part of the value for this attribute. For example:

  ```
  Identity X500DN cn=#my  identity
  value used: cn=#my  identity
  ```

Restriction: When the value contains embedded blanks, you must specify the entire parameter value within the first 1 536 characters of the configuration file line.

Location

ipaddress
: A single IP address that represents the location of the local security endpoint.

  Rule: The IPv6 unspecified address (::0) is not allowed.

ipaddress/prefixLength
: The range of acceptable local security endpoint IP addresses. The *prefixLength* value is the number of unmasked leading bits in the specified IP address and can have a value in the range 0 - 32 for IPv4 addresses and a value in the range 0 - 128 for IPv6 addresses.

  Rule: The IPv6 unspecified address (::0/128) is not allowed.

  Restrictions:

  - You cannot specify a range of IP addresses for a local security endpoint that is referenced by an IpLocalStartAction statement.
  - This value is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

ipaddress-ipaddress

The range of IP addresses that are acceptable local security endpoint addresses.

Rule: The IPv6 unspecified address (::0-::0) is not allowed.

Restrictions:

- You cannot specify a range of IP addresses for a local security endpoint that is referenced by an IpLocalStartAction statement.
- This value is valid only for V1R10 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

Any
The value **Any** is valid only in a host-to-host or host-to-gateway configuration. In the context of a KeyExchangeRule statement, **Any** indicates that any local IPv4 address can represent the location of the local security endpoint for this KeyExchangeRule statement. In the context of an IpLocalStartAction statement, **Any** indicates that the source IPv4 address from the outbound packet (in the case of an on-demand activation), or the LocalIp keyword (in the case of activation based on a LocalDynVpnRule statement) is used as the location of the local security endpoint. **Any** and **Any4** are interchangable values.

Any4
The value of **Any4** is valid only in a host-to-host or host-to-gateway configuration. In the context of a KeyExchangeRule statement, **Any4** indicates that any local IPv4 address can represent the location of the local security endpoint for this KeyExchangeRule statement. In the context of an IpLocalStartAction statement, **Any4** indicates that the source IPv4 address from the outbound packet in the case of an on-demand activation, or the LocalIp keyword (in the case of activation based on a LocalDynVpnRule statement) is used as the location of the local security endpoint.

Any6
A value of **Any6** is valid only in a host-to-host or host-to-gateway configuration. In the context of a KeyExchangeRule statement, Location **Any6** indicates that any local IPv6 address can represent the location of the local security endpoint for this KeyExchangeRule statement. In the context of an IpLocalStartAction statement, Location **Any6** indicates that the source IPv6 address from the outbound packet (in the case of an on-demand activation), or the LocalIp keyword (in the case of activation based on a LocalDynVpnRule statement) is used as the location of the local security endpoint.

If LocalSecurityEndpoint is configured, then the default value is set to **Any4**.

`LocationRef`
The name of a globally defined IPAddr statement that represents the location of the local security endpoint.

`LocationSetRef`
The name of a globally defined IPAddrSet statement that represents the location of the local security endpoints.

Restrictions:

- You cannot specify a range of IP addresses for a local security endpoint that is referenced by an IpLocalStartAction statement.
- This value is valid only for V1R10 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

LocationGroupRef
The name of a globally defined IPAddrGroup statement that represents the location of the local security endpoints

Restrictions:

- You cannot specify a group of IP addresses for a local security endpoint that is referenced by an IpLocalStartAction statement.
- This value is valid only for V1R10 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

RemoteIdentity statement

Use the RemoteIdentity statement to encapsulate remote IKE identity information. This statement defines a singule or wildcard value remote identity for use in negotiation of dynamic VPN tunnels.

Restriction: This statement is valid only for V1R12 and later releases. See “General syntax rules for Policy Agent” on page 933 for details.

Syntax

```
►►──RemoteIdentity──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                    └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬───────────────────────────────────────────────┤
   ├─┤ RemoteIdentity Parameters ├──┤
   └─}──────────────────────────────┘
```

RemoteIdentity Parameters:

```
├──Identity──┬─IpAddr authid───────────────┬───────────────────────────────────────┤
             ├─KeyID ──┬─Ascii authid──┬───┤
             │         ├─Ebcdic authid─┤   │
             │         └─Hex authid────┘   │
             ├─Fqdn authid─────────────────┤
             ├─UserAtFqdn authid───────────┤
             └─X500dn authid───────────────┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this RemoteIdentity statement.

Rule: If this RemoteIdentity statement is not specified as an inline statement, you must specify a *name* value. If you do not specify a name for an inline RemoteIdentity statement, a nonpersistent system name results.

Identity
The identity of a remote security endpoint with which dynamic VPN tunnel negotiations should be allowed. The RemoteIdentity statement supports the following identity types and formats, which can be coded with a wildcard value to indicate a set of remote endpoints:

IpAddr

Indicates that the *authid* value is an IP address, for example: 1.2.3.4 or 1::9. This value can be coded with a wildcard value as a subnet or range.

The following code is a subnet example:

```
1.2.3.0/24 or 1::9/124
```

The following code is a range example:

```
1.2.3.4-1.2.3.100 or 1::0-1::F
```

KeyID

Indicates that the *authid* value is an opaque byte stream. This identity type is intended for use with pre-shared key authentication. The ID value can be specified as an ASCII string, an EBCDIC string, or a hexadecimal string. The maximum length for an ASCII or EBCDIC string is 900 characters. The maximum length for a hexadecimal string is 450 bytes. The hexstring must begin with a 0x.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

Examples:

KeyID Ascii SharedKeyValue

The value is treated as an ASCII string. This specification is valuable if the key ID is defined to the other endpoint as an ASCII string.

KeyID Ebcdic SharedKeyValue

The value is treated as an EBCDIC string.

KeyID Hex 0xC1C2C3F1F2F3

The value is treated as a hexadecimal string.

The ASCII or EBCDIC KeyID value can be defined as a quoted string or a single value.

Rules:

- A quoted string must start and end with a double-quote (").
- A quoted string allows the KeyID value to have embedded blanks for the attribute.
- If KeyID value is not a quoted string then it as treated as a single value.

Results:

- Leading blanks and trailing blanks within the quoted string are removed.
- Within a quoted string, comment indicators, embedded blanks, and additional quotes are treated as part of the value for this attribute.

Restriction: When the value contains embedded blanks, you must specify the entire parameter value within the first 1 536 characters of the configuration file line.

Example KeyID values:

```
Identity KeyID Ascii   ASC # comment"  value used:  ASC
Identity KeyID EBCDIC  EBC comment     value used:  EBC
Identity KeyID ASCII   "ASC 98Z"       value used:  ASC 98Z
Identity KeyID EBCDIC  EBC 98Z"        value used:  EBC
Identity KeyID ASCII   "AsC 98Z        value used:  "AsC
Identity KeyID EBCDIC  "Ebc " " Ebc"   value used:  Ebc " " Ebc
Identity KeyID ASCII   "Asc Asc" "     value used:  Asc Asc"
```

Fqdn Indicates that the *authid* value is a fully qualified domain name or host name. For example, vnet.ibm.com. The maximum length accepted is 1024 characters. The Fqdn value cannot begin or end with a dot (.) and cannot contain consecutive dots.

The Fqdn value can be coded with a wildcard value in the leftmost portion preceding the first period. For example, *.ibm.com is allowed.

The leftmost portion cannot be a partial wildcard value. For example, *net.ibm.com is not allowed.

UserAtFqdn
Indicates that the *authid* value is a user at a fully qualified domain name or host name. The user name cannot contain a blank.

For example, ibm@vnet.ibm.com is allowed. The maximum length accepted is 1024 characters. The UserAtFqdn value cannot begin or end with a dot (.) and cannot contain consecutive dots.

The user portion can be a wildcard value (for example, *@vnet.ibm.com). Alternatively, the leftmost portion of the Fqdn value can be a wildcard value. For example, *.ibm.com is allowed.

X500dn
Indicates that the *authid* value is an X.500 distinguished name (DN). See “LocalSecurityEndpoint statement” on page 1138 for the DN specification.

The leftmost portion of the DN can be a wildcard value. For example, *,OU=endicott,O=ibm,C=US is allowed.

Non-initial RDNs cannot be a wildcard value. For example, CN="John Doe",*,O=ibm,C=US is not allowed.

Rule: You can use comment indicators and embedded blanks as part of the value for this attribute. For example:

```
Identity X500DN cn=#my  identity
value used: cn=#my  identity
```

Restriction: When the value contains embedded blanks, you must specify the entire parameter value within the first 1 536 characters of the configuration file line.

RemoteSecurityEndpoint statement

Use the RemoteSecurityEndpoint statement to encapsulate remote security endpoint IP addresses or hostnames and identity information. This statement defines identity requirements for remote security endpoints with which negotiations for dynamic VPN tunnels are allowed. The statement can also list one or more Certificate Authorities to be used with the allowed security endpoints.

Guideline: The IP address of a remote system is always a public address when the remote security endpoint is behind a NAT device. The NAT device uses the private IP address of the remote security endpoint to choose a public address and replaces it in the IP header.

Syntax

```
►►──RemoteSecurityEndpoint──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├────────►◄
                            └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────────┬─────────────────────────────────────────┤
   ├─┤ RemoteSecurityEndpoint Parameters ├──┤
   └─}──────────────────────────────────────┘
```

RemoteSecurityEndpoint Parameters:

```
├──┬─Identity──┬─IpAddr authid──────────────┬─┬─────────────────────────────────────────►
   │           ├─KeyID ──┬─Ascii authid──┬──┤ │
   │           │         ├─Ebcdic authid─┤  │ │
   │           │         └─Hex authid────┘  │ │
   │           ├─Fqdn authid────────────────┤ │
   │           ├─UserAtFqdn authid──────────┤ │
   │           └─X500dn authid──────────────┘ │
   └─RemoteIdentityRef name───────────────────┘


                                                 ┌──────────────────┐
                                                 ▼                  │
►─┬─Location──Any4───────────────────────────┬───┬────────────────┬─┴─────────────────┤
  ├─Location──┬─ipaddress────────────────────┤   └─CaLabel label──┘
  │           ├─ipaddress/prefixLength─┤     │
  │           ├─ipaddress-ipaddress────┤     │
  │           ├─Any────────────────────┤     │
  │           ├─Any4───────────────────┤     │
  │           └─Any6───────────────────┘     │
  ├─LocationRef name─────────────────────────┤
  ├─LocationSetRef name──────────────────────┤
  └─LocationGroupRef ────────────────────────┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this RemoteSecurityEndpoint statement.

  Rule: If this RemoteSecurityEndpoint statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline RemoteSecurityEndpoint statement, a nonpersistent system name is created.

`Identity`
: The identity of a remote security endpoint with which dynamic VPN tunnel negotiations should be allowed. The RemoteSecurityEndpoint identity supports the same identity types and formats as the LocalSecurityEndpoint identity. In addition, the RemoteSecurityEndpoint identity can be wildcarded to indicate a set of acceptable endpoints.

  The following identity types and formats are supported:

IpAddr

Indicates that the *authid* value is an IP address, for example: 1.2.3.4 or 1::9. This value can be wildcarded as a subnet or range.

The following code is a subnet example:

```
1.2.3.0/24 or 1::9/124
```

The following code is a range example:

```
1.2.3.4-1.2.3.100 or 1::0-1::F
```

KeyID

Indicates that the *authid* value is an opaque byte stream. This identity type is intended for use with pre-shared key authentication. The ID value can be specified as an ASCII string, an EBCDIC string, or a hexadecimal string. The maximum length for an ASCII or EBCDIC string is 900 characters. The maximum length for a hexadecimal string is 450 bytes. The hexstring must begin with a 0x.

Examples:

KeyID Ascii SharedKeyValue
The value is treated as an ASCII string. This specification is valuable if the key ID is defined to the other endpoint as an ASCII string.

KeyID Ebcdic SharedKeyValue
The value is treated as an EBCDIC string.

KeyID Hex 0xC1C2C3F1F2F3
The value is treated as a hexadecimal string.

The ASCII or EBCDIC KeyID value can be defined as a quoted string or a single value.

Rules:

- A quoted string must start and end with a double-quote (").
- A quoted string allows the KeyID value to have embedded blanks for the attribute.
- If KeyID value is not a quoted string then it as treated as a single value.

Results:

- Leading blanks and trailing blanks within the quoted string are removed.
- Within a quoted string, comment indicators, embedded blanks, and additional quotes are treated as part of the value for this attribute.

Restriction: This value is valid only for V1R12 and later releases. See "General syntax rules for Policy Agent" on page 933 for details

Restriction: When the value contains embedded blanks, you must specify the entire parameter value within the first 1 536 characters of the configuration file line.

Example KeyID values:

```
Identity KeyID Ascii   ASC # comment"  value used:  ASC
Identity KeyID EBCDIC  EBC comment     value used:  EBC
Identity KeyID ASCII   "ASC 98Z"       value used:  ASC 98Z
Identity KeyID EBCDIC  EBC 98Z"        value used:  EBC
Identity KeyID ASCII   "AsC 98Z        value used:  "AsC
Identity KeyID EBCDIC  "Ebc " " Ebc"   value used:  Ebc " " Ebc
Identity KeyID ASCII   "Asc Asc" "     value used:  Asc Asc"
```

Fqdn Indicates that the *authid* value is a fully qualified domain name or host name. For example, vnet.ibm.com. The maximum length accepted is 1024 characters. The Fqdn value cannot begin or end with a dot (.), or contain consecutive dots.

The fqdn value can be wildcarded in the leftmost portion preceding the first period. For example, *.ibm.com is allowed.

The leftmost portion cannot be partially wildcarded. For example, *net.ibm.com is not allowed.

UserAtFqdn
Indicates that the *authid* value is a user at a fully qualified domain name or host name. The user name cannot contain a blank.

For example, ibm@vnet.ibm.com. The maximum length accepted is 1024 characters. The UserAtFqdn value cannot begin or end with a dot (.), or contain consecutive dots.

The user portion can be wildcarded. For example, *@vnet.ibm.com. Alternatively, the leftmost portion of the fqdn can be wildcarded. For example, *.ibm.com

X500dn
Indicates that the *authid* value is an X.500 distinguished name (DN). See "LocalSecurityEndpoint statement" on page 1138 for the DN specification.

The leftmost portion of the DN can be wildcarded. For example, *,OU=endicott,O=ibm,C=US is allowed.

Non-initial RDNs cannot be wildcarded. For example, CN="John Doe",*,O=ibm,C=US is not allowed.

Rule: You can use comment indicators and embedded blanks as part of the value for this attribute. For example:

```
Identity X500DN cn=#my  identity
value used: cn=#my  identity
```

Restriction: When the value contains embedded blanks, you must specify the entire parameter value within the first 1 536 characters of the configuration file line.

RemoteIdentityRef
The name of a globally defined RemoteIdentity statement that indicates the identity of a remote security endpoint with which dynamic VPN tunnel negotiations should be allowed.

Restriction: This parameter is valid only for V1R10 and later releases. See"General syntax rules for Policy Agent" on page 933 for details.

Location

ipaddress
A single IP address specification of a remote security endpoint with which dynamic VPN tunnel negotiations should be allowed.

Rule: The IPv6 unspecified address (::0) is not allowed.

ipaddress/prefixLength
A prefix address specification of a range of acceptable remote security endpoint IP addresses. The *prefixLength* value is the number of

unmasked leading bits in the specified IP address and can have a value in the range 0 - 32 for IPv4 addresses and from 0 - 128 for IPv6 addresses.

Rule: The IPv6 unspecified address (::0/128) is not allowed.

ipaddress-ipaddress
A range of IP address specifications of acceptable remote security endpoint addresses for dynamic VPN tunnel negotiations.

Rule: The IPv6 unspecified address (::0-::0) is not allowed.

Any Specifies all IPv4 addresses. **Any** and **Any4** are interchangeable values.

Any4 Specifies all IPv4 addresses.

Any6 Specifies all IPv6 addresses.

Result: If RemoteSecurityEndpoint is configured then the default value is set to **Any4**.

LocationRef
The name of a globally defined IPAddr statement for the remote security endpoint with which dynamic VPN tunnel negotiations should be allowed.

LocationSetRef
The name of a globally defined IPAddrSet statement for the remote security endpoint set with which dynamic VPN tunnel negotiations should be allowed.

LocationGroupRef
The name of a globally defined IPAddrGroup statement for the remote security endpoint group with which dynamic VPN tunnel negotiations should be allowed.

Restrictions:

- You cannot specify a group of IP addresses for a remote security endpoint that is referenced by an IpLocalStartAction statement.
- This parameter is valid only for V1R10 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

CaLabel
Use CaLabel to indicate which certificate authority the remote security endpoint should use when sending a certificate. Multiple instances of this keyword are permitted, indicating that there may be more than one acceptable certificate authority. The remote security endpoint may choose not to honor the request, in which case the negotiation may fail.

label A label identifying a portion of a certificate authority hierarchy.

Rule: When IKED is configured to use local certificate services the label specified on the CaLabel parameter must be the label of a certificate authority certificate on the IKE servers key ring. This label must also be specified on the SupportedCertAuth parameter of the IkeConfig statement. See the description of SupportedCertAuth parameter in "IkeConfig statement" on page 443 for more information. This label identifies a specific certificate authority that the local security endpoint prefers. For example, consider a certificate hierarchy that consists of a Root CA, a subordinate CA X created by the Root CA, and a subordinate CA Y created by CA X.

- If the peers certificate should only be issued by the CA Y, then a CaLabel parameter with the label CA Y should be specified.

- If it is acceptable for the peers certificate to be issued by the CA Y or CA X, then a CaLabel parameter with label CA Y and a second CaLabel parameter with label CA X should be specified.
- If it is acceptable for the peers certificate to be issued by the Root CA, CA Y or CA X, then multiple CaLabel parameters should be specified, one for each of the acceptable certificate authorities.

Result: When IKED is configured to use local certificate services and no CaLabel parameters are specified, the SupportedCertAuth parameter on the IkeConfig statement provides the list of acceptable certificate authorities that the remote security endpoint should use. See the description of SupportedCertAuth parameter in “IkeConfig statement” on page 443 for more information.

Rule: When IKED is configured to use the Network Security Server's certificate services the label specified on the CaLabel parameter must be the label of a certificate authority certificate on the NSSD servers key ring and the stack must be authorized to the EZB.NSSCERT.*sysname.mappedlabelname*.CERTAUTH profile. This label identifies the start of a sub-hierarchy that the local security endpoint prefers. For example, consider a certificate hierarchy that consists of a Root CA, a subordinate CA X created by the root CA, and a subordinate CA Y created by CA X.

- If the peers certificate should only be issued by CA Y then a CaLabel parameter with the label of CA Y should be specified.
- If it is acceptable for the peers certificate to be issued by CA Y or CA X then only a CaLabel parameter with the label of CA X would need to be specified.
- If it is acceptable for the peers certificate to be issued by the Root CA, CA Y or CA X then only a CaLabel parameter with the label of the Root CA would need to be specified.

Result: When IKED is configured to use the Network Security Servers certificate services and no CaLabel parameters are specified, any certificate authority that has a certificate authority certificate on the NSSD keyring to which the stack is authorized is acceptable for the remote security endpoint to use. The EZB.NSSCERT.*sysname.mappedlabelname*.CERTAUTH profile is used to authorize a stack to a certificate authority certificate.

Rule: Comment indicators and embedded blanks are treated as part of the value for this attribute. For example:

```
CaLabel  Root#CA  Certificate
value used:   Root#CA  Certificate
```

Restriction: When the value contains embedded blanks, you must specify the entire value within the first 1 536 characters of the configuration file line.

Policy-based routing policy statements

This topic contains information about the following Routing policy statements:

- “RouteTable statement” on page 1152
- “RoutingAction statement” on page 1163
- “RoutingRule statement” on page 1164

For an example of Routing policy definitions see the pagent_routing.conf file in the /usr/lpp/tcpip/samples/ directory.

RouteTable statement

Use the RouteTable statement to create a table of the routes that can be used to route IP packets based on policy. A RoutingRule statement specifies the characteristics of IP packets and references a RoutingAction statement, which can reference one or more RouteTable statements.

The RouteTable statement is used to create a table of static and dynamic routes. The RouteTable statement is made up of Route entries and DynamicRoutingParms entries. A route entry is used to create a static route in the route table. The syntax for the route entry is compatible with UNIX standards and similar to the syntax for static routes in the TCP/IP profile's BEGINROUTES block. Dynamic routes are added to the route table by OMPROUTE based on the information provided in DynamicRoutingParms entries. IPv6 router advertisement routes are also added to the route table based on the information that is provided in the DynamicRoutingParms entries.

Restrictions:
- A limit of 255 route tables is allowed.
- Duplicate RouteTableRef parameters are not allowed within a RoutingAction statement

The route table can be modified as follows:
- Incoming ICMP and ICMPv6 redirect packets can replace static routes, and can also add routes to the route table.
- The OMPROUTE dynamic routing daemon can replace replaceable static routes, and can add dynamic routes to the route table.
- IPv6 router advertisement routes can be added to the route table based on received IPv6 router advertisements, and can replace replaceable static routes.
- Direct host routes to dynamic XCF addresses on other TCP/IP stacks are added when both of the following conditions are true:
  – The dynamic XCF links to those stacks are active.
  – DynamicXCFRoutes Yes or DynamicXCFRoutes6 Yes is specified on the RouteTable statement.

When a RouteTable statement is updated, the route table in the TCP/IP stack is updated. When a route entry is added, deleted, or updated, the static routes in the route table are updated and routes learned by way of ICMP or ICMPv6 redirect are deleted from the route table. When a DynamicRoutingParms entry is added, deleted, or updated, OMPROUTE updates the dynamic routes in the route table as needed, and any IPv6 router advertisement routes in the route table are updated as needed.

Route precedence is as follows:
1. If a route exists to the destination address (a host route), it is chosen first.
2. For IPv4, if subnet, network, or supernetwork routes exist to the destination, the route with the most specific network mask is chosen second. The most specific network mask is the mask with the most bits on. For IPv6, if prefix routes exist to the destination, the route with the most specific prefix is chosen second.
3. For IPv4, if the destination is a multicast destination and multicast default routes exist, the route with the most specific multicast address is chosen third.
4. Default routes are chosen when no other route exists to a destination.

Rules:

- The RouteTable statement must contain at least one Route entry or one DynamicRoutingParms entry; otherwise, you must specify either DynamicXCFRoutes Yes or DynamicXCFRoutes6 Yes.
- The required parameters for the route entry must be specified in the order shown. The optional parameters can be specified in any order.
- The parameters for the DynamicRoutingParms entry must be specified in the order shown.

Tip: The Options parameters on the route entry can be abbreviated using the same syntax that is used for static routes in the TCP/IP profile's BEGINROUTES block.

Syntax

```
►►──RouteTable──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬───────────────────────────────────────────────┤
   ├─┤ RouteTable Parameters ├──┤
   └─{──────────────────────────┘
```

RouteTable Parameters:

```
   ┌─IgnorePathMtuUpdate No──────────┐  ┌─IgnorePathMtuUpdate6 No──────────┐
├──┴─IgnorePathMtuUpdate──┬─Yes─┬────┴──┴─IgnorePathMtuUpdate6──┬─Yes─┬────┴────────►
                          └─No──┘                               └─No──┘

   ┌─Multipath UseGlobal──────────────┐  ┌─Multipath6 UseGlobal─────────────┐
►──┴─Multipath──┬─PerConnection─┬─────┴──┴─Multipath6──┬─PerConnection─┬───┴──────►
                ├─PerPacket─────┤                      ├─PerPacket─────┤
                ├─Disable───────┤                      ├─Disable───────┤
                └─UseGlobal─────┘                      └─UseGlobal─────┘

   ┌─DynamicXCFRoutes No──────────┐  ┌─DynamicXCFRoutes6 No──────────┐
►──┴─DynamicXCFRoutes──┬─Yes─┬────┴──┴─DynamicXCFRoutes6──┬─Yes─┬───┴────────────►
                       └─No──┘                            └─No──┘

   ┌──────────────────────────────────────────────────────────────────────┐
   │                                                      ┌────────────┐  │
   ▼                                                      ▼            │  │
►──┬────────────────────────────────────────────────────┬───┬──────────┬┴──┴─────►
   └─Route──┤ Destination ├──┤ First Hop ├──┤ Packet Size ├┘ └┤ Options ├┘

   ┌──────────────────────────────────────────────────────┐
   ▼                                                      │
►──┬──────────────────────────────────────────────────────┬┴────────────────────┤
   └─DynamicRoutingParms ──intf_name──┬─IPV4─────────┬────┘
                                      ├─IPV6─────────┤
                                      └─gateway_addr─┘
```

Destination:

```
├──┬─ipaddress/prefixLength─┬──────────────────────────────────────────────┤
   ├─ipaddress──────────────┤
   ├─DEFAULT6───────────────┤
   └─DEFAULT────────────────┘
```

First Hop:

```
├──┬─gateway_addr─┬──intf_name──────────────────────────────────────────────┤
   └─=────────────┘
```

Packet Size:

```
├──MTU──┬─mtu_size────┬──────────────────────────────────────────────────────┤
        └─DEFAULTSIZE─┘
```

Options:

```
   ┌─NOREPLaceable─┐   ┌─MAXImumretransmittime 120.00──┐
├──┼───────────────┼───┼───────────────────────────────┼────────────────────►
   ├─REPLaceable───┤   └─MAXImumretransmittime seconds─┘
   └─NOREPLaceable─┘

   ┌─MINImumretransmittime 0.50─────┐  ┌─ROUNDTRIPGain 0.125─┐
►──┼────────────────────────────────┼──┼─────────────────────┼──────────────►
   └─MINImumretransmittime seconds──┘  └─ROUNDTRIPGain value─┘

   ┌─VARIANCEGain 0.25──┐   ┌─VARIANCEMultiplier 2.00──┐   ┌─DELAYAcks───┐
►──┼────────────────────┼───┼──────────────────────────┼───┼─────────────┼──┤
   └─VARIANCEGain value─┘   └─VARIANCEMultiplier value─┘   ├─NODELAYAcks─┤
                                                           └─DELAYAcks───┘
```

Parameters

name
A string 1 - 8 characters in length specifying the name of this RouteTable statement.

Restriction: Do not specify the values EZBMAIN or ALL (in any combination of upper and lower case letters) for the name value. The name EZBMAIN is reserved for the main route table that is generated by the TCP/IP profile's BEGINROUTES or GATEWAY statements. The name ALL is reserved for use with the PR modifier of the Netstat ROUTE/-r command.

IgnorePathMtuUpdate
Indicates whether IPv4 ICMP Fragmentation Needed messages should be ignored for this route table. When IPv4 path MTU discovery is enabled (PATHMTUDISCOVERY parameter on the IPCONFIG statement in the TCP/IP profile), IPv4 ICMP Fragmentation Needed messages are used to lower the MTU value that is used to send data to a specific IPv4 destination.

No IPv4 ICMP Fragmentation Needed messages should be processed for this route table. This is the default.

Yes IPv4 ICMP Fragmentation Needed messages should be ignored for this route table.

You might want to ignore the path MTU update for a policy-based route table that contains routes that are known to support large path MTU values. If routes in another route table are defined to the same destination or destinations that need a smaller path MTU value, specifying IgnorePathMtuUpdate Yes ensures that a path MTU update that is the result of sending data on a small MTU route does not cause an update to the path MTU for the route in the policy-based route table.

Guideline: The IgnorePathMtuUpdate option is an advanced option. You do not need to set IgnorePathMtuUpdate Yes. If you specify IgnorePathMtuUpdate Yes, path MTU updates are ignored for all IPv4 routes in the route table.

`IgnorePathMtuUpdate6`

Indicates whether IPv6 ICMP Packet Too Big messages should be ignored for this route table. IPv6 ICMP Packet Too Big messages are used to lower the MTU value that is used to send data to a specific IPv6 destination.

No IPv6 ICMP Packet Too Big messages should be processed for this route table. This is the default.

Yes IPv6 ICMP Packet Too Big messages should be ignored for this route table.

You might want to ignore the path MTU update for a policy-based route table that contains routes that are known to support large path MTU values. If routes in another route table are defined to the same destination or destinations that need a smaller path MTU value, specifying IgnorePathMtuUpdate6 Yes ensures that a path MTU update that is the result of sending data on a small MTU route does not cause an update to the path MTU for the route in the policy-based route table.

Guideline: The IgnorePathMtuUpdate6 option is an advanced option. You do not need to set IgnorePathMtuUpdate6 Yes. If you specify IgnorePathMtuUpdate6 Yes, path MTU updates are ignored for all IPv6 routes in the route table.

`Multipath`

Indicates whether the multipath routing selection algorithm is enabled for outbound IPv4 traffic that uses this policy-based route table.

UseGlobal
Use the MULTIPATH or NOMULTIPATH setting from the IPCONFIG statement in the TCP/IP profile to determine multipath processing. This is the default.

PerConnection
Enables the multipath routing selection algorithm for outbound IPv4 traffic that uses this policy-based route table. If multiple equal-cost routes to an IPv4 destination exist in this policy-based route table, a round-robin algorithm is used to select a route. After a route is selected, connection-oriented or connectionless-oriented IP packets using the same association always use the same route, as long as that route is active.

PerPacket

Enables the multipath routing selection algorithm for outbound IPv4 traffic that uses this policy-based route table. If multiple equal-cost routes to an IPv4 destination exist in this policy-based route table, a round-robin algorithm is used to select a route. Connection or connectionless oriented IP packets using the same association do not always use the same route, but do use all possible active routes to the destination.

The PerPacket option should not be used if IP security is enabled on the IPCONFIG statement in the TCP/IP profile. If Multipath PerPacket is specified for a policy-based route table and the route table is installed in a TCP/IP stack with IPv4 security enabled, multipath routing for IPv4 traffic is disabled. The following message is displayed:
```
EZD0028I IPV4 MULTIPATH PERPACKET NOT VALID WITH IPV4 SECURITY -
MULTIPATH SUPPORT DISABLED FOR  ROUTE TABLE table
```

The Netstat ROUTE/-r PR command displays the MultiPath setting `No(Policy)` if multipath routing for IPv4 traffic has been disabled. This occurs because IPv4 security is enabled.

Disable

Disables the multipath routing selection algorithm for outbound IPv4 traffic that uses this policy-based route table. If multiple equal-cost routes to an IPv4 destination exist in this policy-based route table, the first active route that is found is used to send each IP packet to that destination.

`Multipath6`

Indicates whether the multipath routing selection algorithm is enabled for outbound IPv6 traffic that uses this policy-based route table.

UseGlobal

Uses the MULTIPATH or NOMULTIPATH setting from the IPCONFIG6 statement in the TCP/IP profile to determine multipath processing. This is the default.

PerConnection

Enables the multipath routing selection algorithm for outbound IPv6 traffic that uses this policy-based route table. If multiple equal-cost routes to an IPv6 destination exist in this policy-based route table, a round-robin algorithm is used to select a route. After a route is selected, connection-oriented or connectionless-oriented IP packets that use the same association always use the same route, as long as that route is active.

PerPacket

Enables the multipath routing selection algorithm for outbound IPv6 traffic that uses this policy-based route table. If multiple equal-cost routes to an IPV6 destination exist in this policy-based route table, a round-robin algorithm is used to select a route. Connection-oriented or connectionless-oriented IP packets that use the same association do not always use the same route, but use all possible active routes to the destination.

If IP security is enabled on the IPCONFIG6 statement in the TCP/IP profile, do not use the PerPacket option. If Multipath6 PerPacket is specified for a policy-based route table and the route table is installed in a TCP/IP stack with IPv6 security enabled, multipath routing for IPv6 traffic is disabled. The following message is displayed: `EZD0028I`

```
IPV6 MULTIPATH PERPACKET NOT VALID WITH IPV6 SECURITY -
MULTIPATH SUPPORT DISABLED FOR ROUTE TABLE table
```

The Netstat ROUTE/-r PR command displays the MultiPath6 setting `No(Policy)` if multipath routing for IPv6 traffic has been disabled. This occurs because IPv6 security is enabled.

Disable
Disables the multipath routing selection algorithm for outbound IPv6 traffic that uses this policy-based route table. If multiple equal-cost routes to an IPv6 destination exist in this policy-based route table, the first active route that is found is used to send each IP packet to that destination.

DynamicXCFRoutes
Indicates whether direct routes to IPv4 dynamic XCF addresses on other TCP/IP stacks should be added to this route table. The same routes are automatically generated in the main route table when IPv4 dynamic XCF links are active. See the dynamic XCF information in z/OS Communications Server: IP Configuration Guide for information about the dynamic XCF function and the definitions that are automatically generated when IPCONFIG DYNAMICXCF is specified in the TCP/IP profile.

Yes Add direct routes to IPv4 dynamic XCF addresses on other TCP/IP stacks when the dynamic XCF links are active.

No Do not add direct routes to IPv4 dynamic XCF addresses on other TCP/IP stacks. This is the default.

Rule: Duplicate routes are not allowed within a route table. If a statically defined route is a duplicate of a route generated by DynamicXCFRoutes Yes, the statically defined route takes precedence.

DynamicXCFRoutes6
Indicates whether direct routes to IPv6 dynamic XCF addresses on other TCP/IP stacks should be added to this route table. The same routes are automatically generated in the main route table when IPv6 dynamic XCF links are active. See the dynamic XCF information in z/OS Communications Server: IP Configuration Guide for information about the dynamic XCF function and the definitions that are automatically generated when IPCONFIG6 DYNAMICXCF is specified in the TCP/IP profile.

Yes Add direct routes to IPv6 dynamic XCF addresses on other TCP/IP stacks when the dynamic XCF links are active.

No Do not add direct routes to IPv6 dynamic XCF addresses on other TCP/IP stacks. This is the default.

Rule: Duplicate routes are not allowed within a route table. If a statically defined route is a duplicate of a route that the DynamicXCFRoutes6 Yes generates, the statically defined route takes precedence.

Route
A route entry is used to create a static route in the route table.

Restriction: Duplicate routes are not allowed within a RouteTable statement. Duplicate routes have the same destination and first hop specification (interface name and gateway address).

DynamicRoutingParms

A DynamicRoutingParms entry provides parameters for OMPROUTE to use when generating dynamic routes for the route table and for the stack to use when adding IPv6 router advertisement routes to the route table.

Restriction: Duplicate and overlapping DynamicRoutingParms values are not allowed within a RouteTable statement.

In the following example, the DynamicRoutingParms in both Table1 and Table2 are treated as routing policy errors because the DynamicRoutingParms values overlap:

```
RouteTable Table1
{
  DynamicRoutingParms Link1
  DynamicRoutingParms Link1 10.1.2.3
}
RouteTable Table2
{
  DynamicRoutingParms Link2 IPv6
  DynamicRoutingParms Link2 FE80::1:2:3
}
```

In the following example, the DynamicRoutingParms in both Table3 and Table4 are treated as Routing policy errors because the DynamicRoutingParms values are duplicates.

```
RouteTable Table3
{
  DynamicRoutingParms  Link1
  DynamicRoutingParms  Link1
}
RouteTable Table4
{
  DynamicRoutingParms  Link2 FE80::1:2:3
  DynamicRoutingParms  Link2 FE80::1:2:3}
```

ipaddress
The destination address. The address must be a fully qualified IP address.

The DEFAULT or DEFAULT6 keyword in this field specifies a default route. The destination address can be a host, network, subnetwork, supernetwork, or default address. For IPv6, the address cannot be an IPv4-mapped IPv6 address in hexadecimal or dotted decimal format or an IP address with the reserved prefix ::/96. A local address is not valid for the destination address. Multiple routes that have an identical destination can be specified. When multiple routes are specified, all of them are used when multipath is enabled; otherwise, only the first active route that is specified is used.

prefixLength
An integer value that represents the number of bits in the *ipaddress* value that are used to determine the destination address of the route. For an IPv4 destination, the value is in the range 1 - 32. For an IPv6 destination, the value is in the range 1 - 128.

gateway_addr
On a route entry, the *gateway_addr* value is the host IP address of a gateway or router that you can reach directly, and that forwards packets for the destination network or host. The value must be either a fully qualified address or an equal sign (=), which indicates that the messages are routed directly to destinations on that network or directly to that host. A local address is not valid for the gateway address. The equal sign is not supported for a default route entry. For IPv6, the address cannot be an IPv4-mapped IPv6 address in hexadecimal or dotted decimal format or an IP address with the reserved prefix ::/96.

On a DynamicRoutingParms entry, the *gateway_addr* value is one of the following values:

- IPv4

  Indicates that the *intf_name* value that is specified on this DynamicRoutingParms entry is an IPv4 interface. This is the default.
- IPv6

  Indicates that the *intf_name* value that is specified on this DynamicRoutingParms entry is an IPv6 interface.
- The host IP address of a gateway or router that you can reach directly, and that forwards packets for the destination network or host.

  It must be a fully qualified address. A local address is not valid for the gateway address. If an IPv6 address is specified, it must be a link-local address. OMPROUTE uses the IP address and the *intf_name* value to select dynamic routes to be included in this route table. If the *gateway_addr* value is an IPv6 address, the value is used with the *intf_name* value to determine which IPv6 router advertisement routes are added to this route table.

intf_name

The name of an interface as defined on the LINK or INTERFACE statement in the TCP/IP profile.

Restriction: Loopback and VIPA links are not allowed.

On a route entry, the *intf_name* value is the name of the interface through which packets are sent to the specified destination. If an *intf_name* value is specified that is not defined in the TCP/IP profile, the route is created but is not usable until that interface value is defined in the TCP/IP profile.

Tip: Routes that are configured for an undefined interface name are flagged as invalid on a Netstat ROUTE/-r PR display. The flags field includes the letter I.

On a DynamicRoutingParms entry, the *intf_name* value is the name of an interface through which packets can be sent. OMPROUTE uses the *intf_name* value to select dynamic routes to be included in this route table. If *intf_name* is the name of an IPv6 interface, the value is used to determine which IPv6 router advertisement routes are added to this route table. If *gateway_addr* has been specified, then *intf_name* is used in combination with the *gateway_addr* value. If an interface name is specified that is not defined in the TCP/IP profile, no dynamic routes or IPv6 router advertisement routes are created until the interface name is defined in the TCP/IP profile.

MTU *mtu_size*

The maximum transmission unit (MTU) in bytes for the destination. This value can be up to 65535. The keyword DEFAULTSIZE in this field requests that TCP/IP supply a default value of 576 for IPv4 routes and 1280 for IPv6 routes.

See Figure 1 on page 48 for more information about the largest MTU value that each IPv4 link type supports.

See Table 5 on page 142 for more information about the largest MTU value that each IPv4 interface type supports.

See Table 6 on page 142 for more information about the largest MTU value that each IPv6 interface type supports.

Packet size considerations

- The largest *mtu_size* value that z/OS Communications Server can handle varies for different networks. For example, although the largest packet size for the Ethernet protocol is 1500 bytes, the largest packet size for the 802.3 protocol is 1492 bytes.
- The actual packet size is determined by the total network connection.
  - If a locally attached host has a packet size smaller than your packet size, transfers to that host use the smaller size.
  - The TCP maximum segment size for the 3172 Interconnect Controller Program is 4096. Any packet specifications over 4096 are limited by this restriction. For example, if you specified the packet size 4352, the resulting packet size would still only be 4096 plus the header size, for a total packet size of 4132.
- Large packets can be fragmented by intervening gateways for IPv4 only. Fragmenting and reassembling packets is expensive because of high bandwidth use and CPU time. Packets sent through gateways to other networks should use the default size, DEFAULTSIZE, unless one of the following conditions is true:
  - All intervening gateways and networks are known to accept larger packets.
  - IPv4 Path MTU discovery is enabled by using PATHMTUDISCOVERY on the IPCONFIG statement, which results in the TCP/IP stack dynamically learning the maximum MTU for the total network connection. For IPv6, Path MTU discovery is always enabled.
- If this is a CLAW link, the *mtu_size* cannot exceed the *write_size* specified on the corresponding DEVICE statement.
- You cannot specify an MTU value that is smaller than the default MTU size. For IPv4, the default MTU value is 576 and for IPv6 it is 1280.

REPLACEABLE
Indicates that the static route can be replaced by OMPROUTE or IPv6 router advertisements if a dynamic route to the same destination is discovered. This parameter can be abbreviated to REPL.

Restrictions:

- Only one type (replaceable or nonreplaceable) of static route can be defined to the same destination. If multiple route entries are specified to the same destination and the REPLACEABLE or NOREPLACEABLE setting is not the same, it is considered to be a Routing policy error.
- Do not define replaceable static routes to destination addresses that correspond to dynamic VIPAs for which the TCP/IP stack is a sysplex distributor target. This is not validated by Policy Agent.

Tip: You can use the Netstat ROUTE/-r PR ALL RSTAT command to display all replaceable static routes currently configured in policy-based routing tables.

NOREPLACEABLE
Indicates that the static route cannot be replaced by dynamic routes. The static route is always used to reach the destination, regardless of any information that dynamic routes might be available. This is the default behavior. This parameter can be abbreviated NOREPL.

Restriction: Only one type (replaceable or nonreplaceable) of static route can be defined to the same destination. If multiple route entries are specified to the same destination and the REPLACEABLE or NOREPLACEABLE setting is not the same, it is considered to be a Routing policy error.

Retransmission parameter considerations

The parameters listed in this topic affect the TCP retransmit algorithms. When TCP packets are not acknowledged, TCP begins to retransmit these packets at certain time intervals. If these packets are not acknowledged after a specified number of retransmits, TCP aborts the connection. The time interval between retransmissions increases by approximately twice the previous interval until the packets are acknowledged or the connection times out.

The time intervals between retransmissions and the number of times that packets are retransmitted before the connection times out differs for initial connection establishment and for data packets . For initial connection establishment, the initial time interval is set at approximately 3 seconds and the SYN packet is retransmitted 5 times before the connection is timed out. Data packets use a smoothed Round Trip Time (RTT) as the initial time interval, and data packets are retransmitted 15 times before the connection is timed out. All of the remaining parameters listed in this topic affect the data packet retransmission algorithm. Only the MINIMUMRETRANSMITTIME parameter affects the initial connection establishment.

Tip: A new route lookup is performed after every two retransmissions for a data packet. For more information about the route lookup process, see Route selection algorithm in z/OS Communications Server: IP Configuration Guide. Be careful when you design networks with firewalls. A firewall in an alternate routing path can generate a RESET packet for the rerouted data packets, which causes TCP to abort the connection.

The retransmission parameters enable system administrators who are familiar with TCP/IP transmission performance to alter the flow of TCP/IP data packets and acknowledgments. Under normal circumstances, the following occurs:
- TCP typically waits to receive two packets before sending one ACK to acknowledge the data within them.
- When TCP sends a packet, it waits for an acknowledgment. If it times out before getting an acknowledgment, it resends the packet.

Use the following parameters to adjust the retransmission time-out calculations; slower transmission times prevent packets from being resent as quickly:
- MAXIMUMRETRANSMITTIME
- MINIMUMRETRANSMITTIME
- ROUNDTRIPGAIN
- VARIANCEGAIN
- VARIANCEMULTIPLIER
- DELAYACKS
- NODELAYACKS

TCP uses these values in an algorithm called the TCP Retransmission Timeout Calculation, which is described in RFC 793. When you use this calculation, the following occurs:
- A smoothed round trip time (SRTT) and variance (VAR) is updated from the individual RTT derived from each packet acknowledgement.
- The retransmit time for a new packet is set to twice (approximately) the current SRTT value plus the VAR value.
- Each time a packet is retransmitted, the retransmit time value is doubled.

- The actual interval time used for the initial packet and each retransmission is the retransmit time calculated previously, but limited by the configured MINIMUMRETRANSMITTIME and MAXIMUMRETRANSMITTIME values.

DELAYACKS | NODELAYACKS
Controls transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header.

NODELAYACKS
Specifies that an acknowledgment is returned immediately when a packet is received with the PUSH bit on in the TCP header. The NODELAYACKS parameter on the BEGINROUTES, GATEWAY, and RouteTable statements affects only the connections that use this route. Specifying NODELAYACKS on the TCP/IP stack BEGINROUTES or GATEWAY profile statements, or on the Policy Agent RouteTable statement, overrides the specification of the DELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, and TCPCONFIG profile statements.

DELAYACKS
Delays transmission of acknowledgments when a packet is received with the PUSH bit on in the TCP header. The DELAYACKS parameter on the BEGINROUTES, GATEWAY, and RouteTable statements affects only the connections that use this route. This is the default, but you can override the default by specifying the NODELAYACKS parameter on the TCP/IP stack PORT, PORTRANGE, or TCPCONFIG profile statements.

MAXIMUMRETRANSMITTIME
Limits the TCP retransmission interval. Decreasing this value might decrease the total time it takes a connection to timeout. Specifying MAXIMUMRETRANSMITTIME assures that the interval time never exceeds the specified limit. The minimum value that can be specified for MAXIMUMRETRANSMITTIME is 0. The maximum is 999.990. The default is 120 seconds. This parameter does not affect initial connection retransmission.

MINIMUMRETRANSMITTIME
Sets a minimum retransmit interval. Increasing this value might increase the amount of time it takes for TCP to time out a connection. The minimum value that can be specified for MINIMUMRETRANSMITTIME is 0. The maximum is 99.990. The default is 0.5 (500 milliseconds).

ROUNDTRIPGAIN
This value is the percentage of the latest Round Trip Time (RTT) to be applied to the smoothed RTT average. The higher this value, the more influence the latest packet RTT has on the average. The minimum value that can be specified for ROUNDTRIPGAIN is 0. The maximum value is 1.0. The default is 0.125. This parameter does not affect initial connection retransmission.

VARIANCEGAIN
This value is the percentage of the latest RTT variance from the RTT average to be applied to the RTT variance average. The higher this value, the more influence the latest packet's RTT has on the variance average. The minimum value that can be specified for VARIANCEGAIN is 0. The maximum value is 1.0. The default is 0.25. This parameter does not affect initial connection retransmission.

VARIANCEMULTIPLIER
This value is multiplied against the RTT variance in calculating the retransmission interval. The higher this value, the more affect variation in RTT

has on calculating the retransmission interval. The minimum value that can be specified for VARIANCEMULTIPLIER is 0. The maximum value is 99.990. The default is 2. This parameter does not affect initial connection retransmission.

Retransmission parameters

Use the ROUNDTRIPGAIN, VARIANCEGAIN, and VARIANCEMULTIPLIER parameters to instruct TCP how heavily to weigh the most recent behavior of the network versus the long term behavior for updating the SRTT and VAR values. If you specify smaller values for these parameters, TCP attempts to correct for congestion only if the congestion is sustained. With larger values, TCP corrects for congestion more quickly, and the system is more sensitive to variations in network performance. Use the default values (unless your retransmission rate is too high).

Use DELAYACKS to delay the acknowledgments so that they can be combined with data sent to the foreign host.

Results:
- If a HOME entry or INTERFACE is deleted from the TCP/IP profile, all routes for the associated interface become unusable. The routes remain in the route table and become usable again if the HOME entry or INTERFACE is added back to the profile.
- If an interface becomes inactive, then all routes that are associated with that interface are marked inactive by the stack.
- If an interface becomes active, then all static routes that are associated with that interface are marked active by the stack.

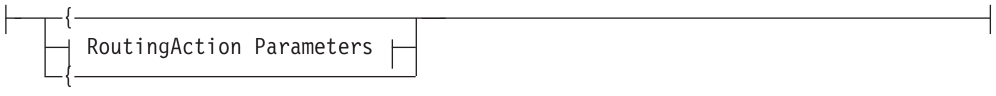
Rules:
- There is no limit to the number of equal-cost multipath routes that can be associated with a single destination.
- Multicast routes can be specified using a host specification. You can also specify multicast network routes or prefix routes.
- A valid host address must contain a nonzero value in the host portion of the address. The host portion of an IPv4 address cannot be all ones, which is considered the broadcast address.
- On an IPv4 route entry, the destination IP address can be either a network or a host IPv4 address, and the *gateway_addr* value must be a host IPv4 address.
- On an IPv6 route entry, the destination IP address can be either a prefix or host IPv6 address, and the *gateway_addr* value must be a host IPv6 address.

RoutingAction statement

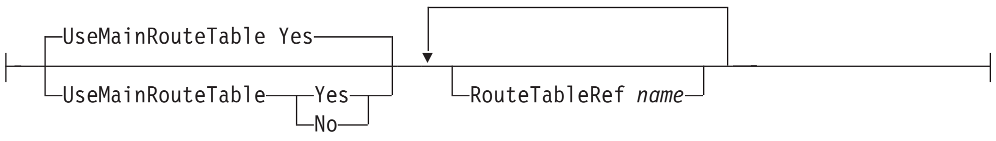
Use the RoutingAction statement to provide an ordered list of RouteTable references and to specify whether the TCP/IP stack's main route table should be used if a usable route is not found in any of the referenced route tables. The stack's main route table is defined in the TCP/IP profile and updated by dynamic routing protocols if configured to do so. A RoutingAction statement can be referenced by a RoutingRule statement.

```
►►──RoutingAction──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:



RoutingAction Parameters:



Parameters

name
: A string 1 - 32 characters in length specifying the name of this RoutingAction statement.

UseMainRouteTable
: Indicates the action taken by the stack when a usable route is not found in any of the referenced route tables.

Yes	When a usable route is not found in any of the referenced route tables, use the stack's main route table to look up a route. This is the default value.
No	When a usable route is not found in any of the referenced route tables, do not use the stack's main route table to look up a route. The packet is not routed.

RouteTableRef
: The name of a globally defined RouteTable statement.

  Restriction: Duplicate RouteTableRef parameters are not allowed in a RoutingAction statement.

Rules:

- If the UseMainRouteTable No value is specified, then at least one RouteTableRef parameter is required.
- A maximum of eight RouteTableRef parameters can be configured for a routing action. Route tables are searched in the order in which the RouteTableRef parameters are specified.

RoutingRule statement

Use the RoutingRule statement to specify characteristics of IP packets that are used to control the route over which the packets can be sent. The RoutingRule statement references a corresponding RoutingAction statement that indicates which route tables to search.

The information provided on the RoutingRule statement defines a routing rule. The routing rule can contain a source IP address, destination IP address, and traffic descriptor specification. The traffic descriptor specification identifies characteristics of IP packets in addition to the IP addresses (for example, source and destination ports). The routing rule can contain a priority and an IpTimeCondition statement specification. An IpTimeCondition statement specification identifies the time period during which the routing rule is in effect.

Restriction: Policy-based routing applies only to TCP and UDP traffic that originates at the TCP/IP stack. Traffic that uses protocols other than TCP and UDP and all traffic that the TCP/IP stack forwards are always routed by using the main route table, even when policy-based routing is in use.

Rules:
- A RoutingRule statement must contain a reference to a globally defined RoutingAction statement.
- If you use default values for the source IP address, destination IP address, and traffic descriptor specifications, the RoutingRule statement applies to all TCP and UDP traffic that originates from this TCP/IP stack.

```
►►──RoutingRule──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{───────────────────────┬────────────────────────────────────────────┤
   ├─┤ RoutingRule Parameters ├─┤
   └─{───────────────────────┘
```

RoutingRule Parameters:

```
   ┌─IpSourceAddr All────────────────────────┐
├──┼─────────────────────────────────────────┼──────────────────────────────►
   ├─IpSourceAddr──┬─ipaddress──────────────┬┤
   │               ├─ipaddress/prefixLength─┤│
   │               ├─ipaddress-ipaddress────┤│
   │               └─All────────────────────┘│
   ├─IpSourceAddrRef name────────────────────┤
   ├─IpSourceAddrSetRef name─────────────────┤
   └─IpSourceAddrGroupRef name───────────────┘

  ┌─IpDestAddr All──────────────────────────┐
►─┼─────────────────────────────────────────┼───────────────────────────────►
  ├─IpDestAddr──┬─ipaddress──────────────┬──┤
  │             ├─ipaddress/prefixLength─┤  │
  │             ├─ipaddress-ipaddress────┤  │
  │             └─All────────────────────┘  │
  ├─IpDestAddrRef name──────────────────────┤
  ├─IpDestAddrSetRef name───────────────────┤
  └─IpDestAddrGroupRef name─────────────────┘

                                         ┌──────────────────────────────┐
                                         ▼                              │
►─┬────────────────────────────────┬───────┬───────────────────────────┬┴──►
  ├─TrafficDescriptor──────────────┤       ├─IpTimeCondition───────────┤
  ├─TrafficDescriptorRef name──────┤       └─IpTimeConditionRef name───┘
  └─TrafficDescriptorGroupRef name─┘
```

```
   ┌─Priority 1─┐
►──┤            ├──RoutingActionRef name──────────────────────────────────────┤
   └─Priority n─┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this RoutingRule statement.

IpSourceAddr
A source address in an outbound IP packet that matches this rule so that the action of the rule can be performed. The default value is All, which indicates that any source address matches this rule.

Guideline: The source IP address for a TCP outbound connection, or for a UDP outbound packet, can be influenced by a number of configuration and application options. See the source IP address information in z/OS Communications Server: IP Configuration Guide for the hierarchy of ways that the source IP address of an outbound packet is determined. For the following source IP address selection methods, a route lookup is needed to determine the source IP address:

- SOURCEVIPA: Static VIPA address from the HOME list (IPv4 interface defined with the LINK statement) or from the SOURCEVIPAINTERFACE parameter (IPv4 or IPv6 interface defined with the INTERFACE statement)
- HOME: IP address of the interface over which the packet is sent

Do not use the IpSourceAddr value to select traffic that relies on these methods to select its source IP address. At the time that route lookup is performed, the source IP address has not yet been selected.

All Any source IP address matches this rule.

ipaddress
A single IP address.

ipaddress/prefixLength
A prefix address specification. The *prefixLength* value is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP packet matches this condition if its unmasked bits are identical to the defined unmasked bits.

Results:

- 0.0.0.0/0 indicates that any IPv4 source address matches this rule.
- ::/0 indicates that any IPv6 source address matches this rule.

ipaddress-ipaddress
A range of IP addresses.

Restriction: If the IP address is an IPv6 address, it cannot be an IPv4-mapped IPv6 address in hexadecimal or dotted decimal format or an IP address with the reserved prefix ::/96. If the IPv6 address is one of these types, an error message is logged.

IpSourceAddrRef
The name of a globally defined IpAddr statement that is used for the source IP address specification.

IpSourceAddrSetRef
The name of a globally defined IpAddrSet statement that is used for the source IP address prefix or range specification.

IpSourceAddrGroupRef
The name of a globally defined IpAddrGroup statement that is used for the source IP address specification.

IpDestAddr
A destination address in an outbound IP packet that matches this rule so that the action of the rule can be performed. The default is All, which indicates that any destination address matches this rule.

All Any destination IP address matches this rule.

ipaddress
A destination IP address.

ipaddress/prefixLength
A prefix address specification. The *prefixLength* value is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and 0 - 128 for IPv6 addresses. An IP packet matches this condition if its unmasked bits are identical to the defined unmasked bits.

Results:

- 0.0.0.0/0 indicates that any IPv4 destination address matches this rule.
- ::/0 indicates that any IPv6 destination address matches this rule.

ipaddress-ipaddress
A range of IP addresses.

Restriction: If the IP address is an IPv6 address, it cannot be an IPv4-mapped IPv6 address in hexadecimal or dotted decimal format or an IP address with the reserved prefix ::/96. If the IPv6 address is one of these types, an error message is logged.

IpDestAddrRef
The name of a globally defined IpAddr statement that is used for the destination IP address specification.

IpDestAddrSetRef
The name of a globally defined IpAddrSet statement that is used for the destination IP address prefix or range specification.

IpDestAddrGroupRef
The name of a globally defined IpAddrGroup statement that is used for the destination IP address specification.

TrafficDescriptor
An inline specification of a TrafficDescriptor statement.

TrafficDescriptorRef
The name of a globally defined TrafficDescriptor statement.

TrafficDescriptorGroupRef
The name of a globally defined TrafficDescriptorGroup statement.

IpTimeCondition
An inline specification of an IpTimeCondition statement. There is a limit of 25 IpTimeCondition specifications in the RoutingRule statement.

IpTimeConditionRef
The name of a globally defined IpTimeCondition statement. There is a limit of 25 IpTimeCondition references in the RoutingRule statement.

Priority
This is an integer value in the range 1 - 2000000000 representing the priority associated with the rule.

Restriction: Only one rule is mapped to a route request. Rules are searched for a match starting at the highest priority, so if multiple rules could possibly be matched for a given route request, the rule with the highest priority gets matched first. If multiple rules of the same priority match, the rule mapped is difficult to predict. If this attribute is not specified, the default priority is 1.

Guideline: When setting the priority for multiple rules, do not set the priority as a sequential value, for example 2, 3 ,4, and 5. Instead, set the priority to provide space to change the priority or to insert additional rules, such that the rule would be preferred before another rule, without duplicating a priority. For example, the priorities could be configured as 20, 30, 40, and 50.

RoutingActionRef
The name of a globally defined RoutingAction statement.

QoS policy statements

This topic contains information about the following QoS policy statements:
- "PolicyAction statement"
- "PolicyRule statement" on page 1176
- "ServiceCategories statement" on page 1183
- "ServicePolicyRules statement" on page 1187

PolicyAction statement

Use the PolicyAction statement to specify the type of service a flow of IP packets (for example, from a TCP connection, or UDP data) should receive end-to-end as they traverse the network. PolicyAction can be repeated with each having a different name so that they can be referenced later.

This statement defines a Version 2 policy action.

Syntax

```
►►──PolicyAction──name──┤ Place Braces and Parameters on Separate Lines ├──────────►◄
```

Place Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ PolicyAction Parameters ├─┤
   └─}──────────────────────────┘
```

PolicyAction Parameters:

```
   ┌─PolicyScope ──Both──────────┐
├──┼─────────────────────────────┼──┬──────────────────────────────────────┬──►
   └─PolicyScope──┬─DataTraffic─┬┘  │ ┌──────────────────────────────────┐ │
                  ├─RSVP────────┤   │ ▼                                  │ │
                  └─Both────────┘   └───OutboundInterface ──address──────┴─┘

                                                 ┌─OutgoingTOS ──0─┐
►──┬───────────────────┬──┬───────────────────┬──┼─────────────────┼──────────►
   └─MaxRate ──Kbps────┘  └─MinRate ──Kbps────┘  └─OutgoingTOS ──n─┘

►──┬──────────────────────────────┬──┬──────────────────────────┬─────────────►
   └─MaxDelay ──milliseconds──────┘  └─MaxConnections ──value───┘

   ┌─FlowServiceType ──ControlledLoad──────┐
►──┼───────────────────────────────────────┼──┬─────────────────────────────┬─►
   └─FlowServiceType──┬─ControlledLoad─┬───┘  └─MaxRatePerFlow ──Kbps───────┘
                      └─Guaranteed─────┘

►──┬─────────────────────────────────┬──┬──────────────────┬──────────────────►
   └─MaxTokenBucketPerFlow ──Kbps────┘  └─MaxFlows ──n─────┘

   ┌─Permission ──Allowed──────┐
►──┼───────────────────────────┼──┬───────────────────────────────────┬───────►
   └─Permission──┬─Allowed─┬───┘  └─DiffServInProfileRate ──Kbps──────┘
                 └─Blocked─┘

                                             ┌─DiffServInProfileTokenBucket ──100─┐
►──┬──────────────────────────────────────┬──┼────────────────────────────────────┼──►
   └─DiffServInProfilePeakRate ──Kbps─────┘  └─DiffServInProfileTokenBucket ──Kb──┘

►──┬──────────────────────────────────────────┬───────────────────────────────►
   └─DiffServInProfileMaxPacketSize ──Kbps────┘

   ┌─DiffServExcessTrafficTreatment ──BestEffort──────┐
►──┼──────────────────────────────────────────────────┼───────────────────────►
   └─DiffServExcessTrafficTreatment──┬─Drop───────┬───┘
                                     └─BestEffort─┘

   ┌─DiffServOutProfileTransmittedTOSByte ──0─┐
►──┼──────────────────────────────────────────┼───────────────────────────────┤
   └─DiffServOutProfileTransmittedTOSByte ──n─┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this policy action.

PolicyScope
Indicates the scope of this PolicyAction. The following values are allowed:

- DataTraffic indicates the scope is Differentiated Services.
- RSVP indicates the scope is Integrated Services (for example, RSVP).
- Both indicates the scope is both DataTraffic + RSVP (this is the default).

Certain attributes of the policy action are used only with certain scope values, as follows:

RSVP FlowServiceType, MaxRatePerFlow, MaxTokenBucketPerFlow, MaxFlows

DataTraffic
All other attributes (Permission applies to all scope values)

When the scope value is specified as Both, both RSVP and DataTraffic attributes can be specified, but the attributes are only applied to the appropriate scope.

OutboundInterface
Specifies an outbound interface used for sysplex distributor distributing stack. Incoming connection requests can be distributed to different target stacks within the sysplex by the sysplex distributor distributing stack based on VIPADIST statements (which define DXCF interfaces) defined for the corresponding distributing stack.

This attribute selects the DXCF interfaces that are available for the incoming connection request that maps to this policy. You can specify IPv4 and IPv6 addresses. You can specify up to 32 instances of this attribute. The value 0 for IPv4 or :: for IPv6 can be specified for the interface, which indicates to the sysplex distributor distributing stack that if it cannot distribute the request to a target stack on one of the other specified interfaces, then the request can be distributed to any of the other eligible target stacks.

For example, suppose 5 target stacks are defined by VIPADIST statements (1.1.1.1 - 5.5.5.5), and 3 interfaces are defined using the OutboundInterface attribute (1.1.1.1, 2.2.2.2, and 0.0.0.0). If an incoming request cannot be distributed to either 1.1.1.1 or 2.2.2.2, then the specification of the 0 interface indicates that the request should be distributed to any of the remaining stacks (3.3.3.3 - 5.5.5.5) that are eligible to service the request. The PolicyScope attribute must specify either DataTraffic or Both to define interfaces using this attribute.

Result: If OutboundInterface specifies only one type of address (IPv4 or IPv6), then inbound connections of the other type is distributed to all available targets. For example, if only IPv6 addresses are specified for OutboundInterface, then incoming connections to IPv4 DVIPAs are not restricted by OutboundInterface; instead, they are distributed to all available IPv4 targets.

Rules:

- If the IP address is IPv6, it cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these two types, an error message is logged.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

MaxRate
An integer value representing the maximum rate in kilobits per second (Kbps) allowed for traffic in this service class. This attribute is valid only for TCP. If not specified or specified as 0, there is no enforcement of the maximum rate of a connection by the local host. If a number other than 0 is specified, each TCP connection that is mapped to this PolicyAction has its rate limited to this MaxRate. Enforcement of the MaxRate is performed by the TCP/IP stack by adjusting the TCP congestion window based on the connection round-trip time (the rate is obtained by taking the congestion window and dividing it by the round-trip time; note the units, for example, byte versus bit, second versus

millisecond). Because the minimum of the congestion window is one TCP segment size, the minimum of the MaxRate that can be enforced is one TCP segment over the round-trip time. If a TCP connection has a very small round-trip delay and traverses over a very high bandwidth network (for example, Gbit Ethernet LAN), the minimum rate that this TCP connection can send (one segment per round-trip time) can be high. Therefore, users and network administrators need to know their network characteristics when setting this MaxRate; it might not be enforceable if the minimum TCP rate (for example, one segment over round-trip time) already exceeds this specified MaxRate. As noted, TCP segment size can play a role in this TCP minimum rate; for example, for a given round-trip delay, the larger the segment size, the higher the minimum TCP rate. There are different factors that can affect the TCP segment size, for example, the local MTU size definition, the Path MTU discovery flow (this mechanism is used to discover the maximum MTU size that can be sent into the network without resulting in IP fragmentation), the receivers maximum segment size, and so on.

MinRate
An integer value representing the minimum rate or throughput (Kbps) allowed for traffic in this service class. This attribute is valid only for TCP. If not specified or specified as 0, there is no enforcement on the minimum rate of a connection by the local host. If a number other than 0 is specified, the rate for any TCP connection that is mapped to this PolicyAction does not fall below this MinRate, unless the network is really congested and by maintaining the minimum rate the network throughput might collapse. Enforcement of the MinRate is performed by the TCP/IP stack by manipulating the congestion window over the connection round-trip time. Unlike the enforcement of MaxRate, if TCP minimum rate due to the segment size or the round-trip time, or both, is already high, and the specified MinRate is already below this rate, it is not necessary for the TCP/IP stack to enforce the MinRate.

OutgoingTOS
Eight bits, left-aligned, representing the ToS or Traffic Class value of outbound traffic belonging to this service class. The default is 0.

Tip: An outbound packet with a ToS or Traffic Class value that consists of zeros enables prioritizing outbound OSA-Express data using the WorkLoad Manager service class importance level. This function is enabled with the WLMPriorityQ parameter. For more information about Workload Manager provided-priorities, see prioritizing outbound OSA-Express data using the WorkLoad Manager service class importance level in z/OS Communications Server: IP Configuration Guide. When WLMPriorityQ is enabled, specify an OutgoingTOS value other than 0 if you want to prevent the assignment of QDIO priority based on the WorkLoad Manager service class importance level.

MaxDelay
An integer value representing the maximum delay (in milliseconds) allowed for traffic in this service class. This attribute is valid only for TCP. The TCP/IP stack does not enforce this delay.

Result: This parameter is no longer supported and is ignored.

MaxConnections
An integer value representing the maximum number of end-to-end connections at any instant in time. This attribute is valid only with TCP. It places a limit on the number of TCP connections mapped to this PolicyAction that can be active at a time. If there is a request for a new TCP connection that maps to this PolicyAction and this limit is exceeded, the connection request is rejected. The default is that there is no policy limit. The MaxConnections attribute is

enforced by the TCP/IP stack. If the connection request is sent by a remote client, a TCP RST segment is returned to notify the client that the connection is refused. The number of rejected connections is maintained and can be retrieved with the **netstat** command using the **-j** option. If the connection request is sent by an application in the local host (for example, using a connect socket call), a return code of permission denied is returned.

Restriction: This attribute only affects new connection requests, not already active connections. For example, if a policy is activated that limits the maximum number of connections to 10, but 15 connections already existed for traffic that maps to the policy rule, then only 10 of the existing connections are mapped to the policy and no new connections are accepted. However, the five other existing connections over the limit remain active and unmapped by the policy.

FlowServiceType

Limits the Type of Service being requested by RSVP applications. Valid values are ControlledLoad (the default) and Guaranteed. Guaranteed service is considered to be *greater than* ControlledLoad service. If ControlledLoad service is specified, and an application requests Guaranteed, the requested service is downgraded to ControlledLoad. To allow RSVP applications to request Guaranteed service, specify Guaranteed for this parameter. All RSVP parameters, FlowServiceType, MaxRatePerFlow, MaxTokenBucketPerFlow, and MaxFlows, are enforced by the RSVP daemon application and not by the TCP/IP stack. The TCP/IP stack, however, maintains traffic statistics of RSVP policies, which can be retrieved with the **netstat** command with option **-j.**

MaxRatePerFlow

Specifies the maximum rate in kilobits per second for RSVP flows. RSVP reservations are based on a traffic specification (Tspec) from the sending application. The Tspec consists of the following values:

- *r* is the token bucket rate in bytes per second.
- *b* is the token bucket depth in bytes.
- *p* is the peak rate in bytes per second.
- *m* is the minimum packet size in bytes.
- *M* is the maximum packet size (MTU) in bytes.

Use this parameter to limit the *r* value of the Tspec. If an RSVP sender application requests a Tspec *r* value larger than this parameter, the request is downgraded to this parameter value.

RSVP receiving applications also specify a resource specification (Rspec) when using Guaranteed service, as part of the reservation request. The Rspec consists of the following values:

- *R* is the rate in bytes per second.
- *S* is the slack term in microseconds.

This parameter is also used to limit the *R* value of the Rspec for reservation requests from RSVP receiver applications using Guaranteed service.

Guideline: This parameter is specified in kilobits per second, while the Tspec and Rspec use bytes per second. To arrive at a compatible specification, multiply the desired Tspec or Rspec value by 8, then divide by 1000. For example, to specify a Tspec r value of 500000 bytes per second, specify a MaxRatePerFlow value of 4000 (500000 * 8 / 1000 = 4000).

The default for this parameter is a system defined maximum.

MaxTokenBucketPerFlow
Specifies the maximum token bucket size in kilobits per second for RSVP flows. RSVP reservations are based on a traffic specification (Tspec) from the sending application. The Tspec consists of the following values:
- *r* is the token bucket rate in bytes per second.
- *b* is the token bucket depth in bytes.
- *p* is the peak rate in bytes per second.
- *m* is the minimum packet size in bytes.
- *M* is the maximum packet size (MTU) in bytes.

This parameter limits the *b* value of the Tspec. If an RSVP sender application requests a Tspec *b* value larger than this parameter, the request is downgraded to this parameter value.

Guideline: This parameter is specified in kilobits, while the Tspec uses bytes. To arrive at a compatible specification, multiply the desired Tspec value by 8, then divide by 1000. For example, to specify a Tspec b value of 75000 bytes, specify a MaxTokenBucketPerFlow value of 600 (75000 * 8 / 1000 = 600).

The default for this parameter is a system defined maximum.

MaxFlows
Specifies the maximum number of reserved flows allowed for RSVP applications. The default is no limit on the number of reserved flows.

Permission
Indicates whether packets belonging to this policy rule should be discarded or allowed to proceed. Valid values are Allowed and Blocked. The default is Allowed.

DiffServInProfileRate
Specifies the mean rate at which traffic belonging to the corresponding policy must be policed. It is a Kbps value and must be policed in kilobits per second (Kbps). The default value is 0, meaning no policing mechanism is enforced. The DiffServ parameters are enforced by the TCP/IP stack. Statistics regarding in-profile byte count can be retrieved using the **netstat** command with option **-j**. This in-profile count can be used to calculate the amount of traffic sent out of profile. The in-profile count should be equal to or less than the total transmitted byte count unless the count wraps.

Unlike MaxRate/MinRate, which applies on a per TCP connection basis, these DiffServ parameters apply to aggregated flows (multiple TCP connections can be mapped to a single policy action). Also, it is important to note that when DiffServ parameters are enforced against TCP traffic, the TCP minimum rate determines whether the DiffServ parameters are enforceable, as described in the attribute MaxRate. This is due to an optimization provision where TCP is forced to slow down when it attempts to send beyond the committed bandwidth specified with DiffServ parameters in the policy action with DiffServExcessTrafficTreatment specified as Drop. TCP cannot slow down beyond the TCP minimum rate, even if a violation occurs.

This rate that is used to generate tokens in the token bucket traffic policing mechanism, but it is not necessarily the user/application generated traffic rate.If this attribute is a nonzero value, the DiffServInProfileTokenBucket value must also be nonzero.

Guideline: This parameter is used by a token bucket mechanism to control the outbound traffic.

DiffServInProfilePeakRate
Specifies the peak rate that traffic belonging to the corresponding policy must be policed. It is a Kbps value and must be policed in kilobits per second (Kbps). The default is 0, which means no policing mechanism is enforced against the peak rate if DiffServInProfileRate is nonzero. When nonzero, it must not be less than that of the DiffServInProfileRate. If this attribute is nonzero, DiffServInProfileRate and DiffServInProfileMaxPacketSize must also be nonzero.

A token bucket mechanism used this parameter to control the outbound traffic.

DiffServInProfileTokenBucket
Specifies the maximum burst size that traffic belonging to the corresponding policy must be policed. It is a kilobits value and must be policed in kilobits (Kb). The default is 100 if DiffServInProfileRate is not 0. The DiffServInProfileTokenBucket attribute is used only when the policy action also uses the DiffServInProfileRate attribute.

A token bucket mechanism used this parameter to control the outbound traffic.

DiffServInProfileMaxPacketSize
Specifies the maximum packet size of traffic belonging to the corresponding policy. Its value is used to police traffic against the peak rate. It is a kilobits value with corresponding policy, in kilobits (Kb). The default is 100 if DiffServInProfilePeakRate is not 0.

Guideline: Due to blocking in z/OS Communications Server, multiple packets tend to be sent back to back. If the maximum packet size is set to the size of one packet, traffic exceeds the peak rate, and those packets are sent as out of profile packets (either with a different ToS or Traffic Class value or dropped) if peak rate enforcement is in effect. To prevent this, the attribute must be set in multiples of the maximum packet size or equal to the token bucket size.

DiffServExcessTrafficTreatment

Specifies what action to take when traffic exceeds its profile. Two values can be specified with this attribute:
- Drop
- BestEffort

The default is BestEffort. These are described directly below.

When the DiffServExcessTrafficTreatment is Drop and the corresponding policy is defined for TCP traffic, z/OS Communications Server optimizes performance by simulating the TCP packet drop and reducing the TCP transmit rate in order to force the outbound traffic to conform to the policy defined bandwidth. This means that the TCP packets that result in excess traffic are transmitted, but the corresponding TCP connections are forced to slow down immediately (by half, which is the TCP behavior under packet loss). This helps avoid retransmissions and prevents further excess traffic. If the policy is defined for UDP, because there is no slowdown mechanism in UDP as in TCP, excess traffic is discarded as specified in the policy definition.

When the DiffServExcessTrafficTreatment is BestEffort, the excess packets are still sent; however, they are sent with the ToS or Traffic Class value specified on DiffServOutProfileTransmittedTOSByte.

DiffServOutProfileTransmittedTOSByte
Specifies the ToS/DS or Traffic Class value to send with the excess traffic (if action is to send excess traffic as best effort instead of dropping).

The normal in profile ToS or Traffic Class value comes from the current OutgoingTOS attribute. This value is specified as a string of eight 1s and 0s. The default is 00000000.

Tip: An outbound packet with a ToS or Traffic Class value that consists of zeros enables prioritizing outbound OSA-Express data using the WorkLoad Manager service class importance level. This function is enabled with the WLMPriorityQ parameter. For more information about Workload Manager provided-priorities, see prioritizing outbound OSA-Express data using the WorkLoad Manager service class importance level in z/OS Communications Server: IP Configuration Guide. When WLMPriorityQ is enabled, specify a DiffServOutProfileTransmittedTOSByte value othle er than 0 if you want to prevent the assignment of QDIO priority based on the WorkLoad Manager service class importance level.

Table 76 provides a mapping of PolicyAction statement parameters to LDAP object classes and attributes.

Table 76. PolicyAction mapping to LDAP

PolicyAction statement parameter	**Object class**	**LDAP attribute**
DiffServExcessTraffic Treatment	ibm-serviceCategoriesAuxClass	ibm-diffServExcessTrafficTreatment
DiffServInProfile MaxPacketSize	ibm-serviceCategoriesAuxClass	ibm-diffServInProfileMaxPacketSize
DiffServInProfile PeakRate	ibm-serviceCategoriesAuxClass	ibm-diffServInProfilePeakRate
DiffServInProfile TokenBucket	ibm-serviceCategoriesAuxClass	ibm-diffServInProfileTokenBucket
DiffServInProfileRate	ibm-serviceCategoriesAuxClass	ibm-diffServInProfileRate
DiffServOutProfile TransmittedTOSByte	ibm-serviceCategoriesAuxClass	ibm-diffServOutProfileTransmittedTOSByte
FlowServiceType	ibm-serviceCategoriesAuxClass	ibm-flowServiceType
MaxConnections	ibm-serviceCategoriesAuxClass	ibm-maxConnections
MaxDelay	ibm-serviceCategoriesAuxClass	ibm-maxDelay
MaxFlows	ibm-serviceCategoriesAuxClass	ibm-maxFlows
MaxRate	ibm-serviceCategoriesAuxClass	ibm-maxRate
MaxRatePerFlow	ibm-serviceCategoriesAuxClass	ibm-maxRatePerFlow
MaxTokenBucketPerFlow	ibm-serviceCategoriesAuxClass	ibm-maxTokenBucketPerFlow
MinRate	ibm-serviceCategoriesAuxClass	ibm-minRate
OutboundInterface	ibm-serviceCategoriesAuxClass	ibm-interface
OutgoingTOS	ibm-serviceCategoriesAuxClass	ibm-outgoingTOS
Permission	ibm-serviceCategoriesAuxClass	ibm-Permission
PolicyScope	ibm-serviceCategoriesAuxClass	ibm-policyScope

Examples

For an example of the PolicyAction statement, see /usr/lpp/tcpip/samples/pagent.conf.

PolicyRule statement

Use the PolicyRule statement to specify characteristics of IP packets that are used to map to a corresponding policy action. It defines a set of IP datagrams that should receive a particular service.

Restriction: This statement defines a Version 2 policy rule.

Syntax

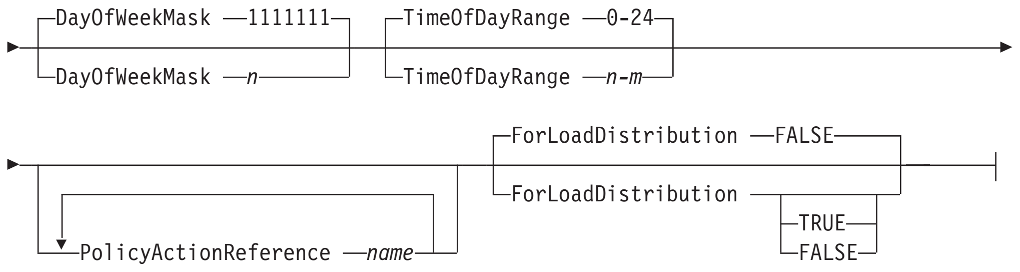
```
►►──PolicyRule──name──┤ Place Braces and Parameters on Separate Lines ├──────────►◄
```

Place Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ PolicyRule Parameters ├─┤
   └─}──────────────────────────┘
```

PolicyRule Parameters:

```
                              ┌─SourceAddressRange all──────────────────┐
├──┬──────────────────────────┬┼─────────────────────────────────────────┼──────►
   └─PolicyRulePriority ──n───┘└─SourceAddressRange ──address address──┘

   ┌─DestinationAddressRange all──────────────────┐ ┌─SourcePortRange ──all─┐
►──┼───────────────────────────────────────────────┼─┼────────────────────────┼──►
   └─DestinationAddressRange ──address address──┘ └─SourcePortRange ──n n─┘

   ┌─DestinationPortRange ──all─┐ ┌─ProtocolNumberRange ──all─┐
►──┼────────────────────────────┼─┼───────────────────────────┼──────────────────►
   └─DestinationPortRange ──n n─┘ └─ProtocolNumberRange ──n───┘

   ┌─InboundInterface ──all─┐ ┌─OutboundInterface ──all─┐
►──┼────────────────────────┼─┼─────────────────────────┼────────────────────────►
   └─InboundInterface ──n───┘ └─OutboundInterface ──n───┘

   ┌─ApplicationName ──all──┐
►──┼────────────────────────┼─┬────────────────────────────────┬─────────────────►
   └─ApplicationName ──name─┘ └─ApplicationData ──string───────┘

►──┬──────────────────────────┬─┬──────────────────────────────┬─────────────────►
   └─ApplicationPriority ──n──┘ └─ConditionTimeRange ──range───┘

   ┌─MonthOfYearMask ──111111111111─┐ ┌─DayOfMonthMask ──31 n's─┐
►──┼────────────────────────────────┼─┼─────────────────────────┼────────────────►
   └─MonthOfYearMask ──n────────────┘ └─DayOfMonthMask ──62 n's─┘
```



Parameters

name
: A string 1 - 32 characters in length specifying the name of this policy rule.

PolicyRulePriority
: PolicyRulePriority specifies the location of the PolicyRule entry in the PolicyRule list. This is an integer type field. Rules are searched for a match starting at the highest priority, so if multiple rules could possibly be matched for a given set of traffic, the rule with the highest priority gets matched first. If multiple rules have the same priority, then the rule with the greatest number of attributes specified gets matched first. If the match criteria is equal, the rule that gets mapped is unpredictable. Only one policy is ever mapped, per PolicyScope attribute. The maximum value for this attribute is 2000000000. If this attribute is specified, the computed priority of the rule is the specified value plus 100. If this attribute is not specified, the computed priority of the rule is determined by the number of selection criteria specified, but is always less than 100. The higher the number defined, the higher the assigned priority.

SourceAddressRange
: Specifies the source addresses of the sender of the traffic flow. The destination of the data can be the client or the server. For TCP connections, the destination of the connection is the client. For inbound connections or traffic, the source is the remote device. For outbound connections or traffic, the source is this host. Both IPv4 and IPv6 addresses can be specified.

  Rules:

  - Include a blank or a dash (-) as a delimiter.
  - If the IP address is IPv6, it cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these two types, an error message is logged.
  - IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

  When the source address range is specified on an LDAP server using the syntax that means *all local addresses*, loopback and loopback-like traffic (for example, otracert from and to a local address), are not mapped due to performance reasons. However, the interface attribute can be specified in addition to the source address to accomplish this mapping.

DestinationAddressRange
: Specifies the destination addresses of the receiver of the traffic flow. The destination of the data might be the client or the server. For inbound connections or traffic, the destination of the connection is this host. For outbound connections or traffic, the destination of the connection is the remote device. Both IPv4 and IPv6 addresses can be specified.

Rules:

- Include a blank or a dash (-) as a delimiter.
- If the IP address is IPv6, it cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these two types, an error message is logged.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

When the destination address range is specified on an LDAP server using the syntax that means *all local addresses*, loopback and loopback-like traffic (for example, otracert from and to a local address), it are not mapped due to performance reasons. However, the interface attribute can be specified in addition to the destination address to accomplish this mapping.

`SourcePortRange`
The source port range. This field consists of two port numbers, separated by a space, where the first port number is less than or equal to the second port number. The default is 0, which is all inclusive. The source of the data can be the client or the server. For inbound connections or traffic, the source is the remote device. For outbound connections or traffic, the source is this host.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

`DestinationPortRange`
The destination port range. This field consists of two port numbers, separated by a space, where the first port number is less than or equal to the second port number. The default is 0, which is all inclusive. The destination of the data can be the client or the server. For inbound connections or traffic, the destination is this host. For outbound connections or traffic, the destination is the remote device.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

`ProtocolNumberRange`
This attribute specifies the protocol range for which this policy rule applies. The format is i1:i2, where i2 >=i1. The maximum value for this attribute is 255. The minimum value is 0, and the default is all protocols. The default and minimum value is 0 and designates all protocols.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

`InboundInterface`
This attribute specifies the inbound local IP subnet for which this policy rule applies. This can be an IPv4 address or an interface name. The default is all interfaces. If an interface name is specified, it must match a name specified on one of the following statements in the TCP/IP profile:

- LINK statement for an IPv4 interface
- INTERFACE statement for an IPv4 or IPv6 interface

Rules:

- InboundInterface and OutboundInterface attributes should not be specified for the same rule, because that would imply a function that is provided by a router.
- The IPv4 address or interface that is defined must be a physical IP address or a physical device, not a virtual device.

`OutboundInterface`
This attribute specifies the outbound local IP subnet for which this policy rule

applies. This can be an IPv4 address or an interface name. The default is all interfaces. If an interface name is specified, it must match a name specified on one of the following statements in the TCP/IP profile:

- LINK statement for an IPv4 interface
- INTERFACE statement for an IPv4 or IPv6 interface

Rules:

- InboundInterface and OutboundInterface attributes should not be specified for the same rule, because that would imply a function that is provided by a router.
- The IPv4 address or interface that is defined must be a physical IP address or a physical device, not a virtual device.

ApplicationName

ApplicationName is a field of type string (up to eight characters) that specifies the job name of the application. Names longer than eight characters are truncated. A trailing asterisk indicates a wildcard specification. For example, if FTPD* is specified, job names of FTPD and FTPD1 match. The application name maps to the sending application for outbound data, and to the receiving application name for inbound data. The name specified here is not case sensitive, and is translated to uppercase before being compared to application names.

The default is all applications.

ApplicationData

This string field of up to 128 characters specifies the application selector data (for example, a URI for the Internet). Strings longer than 128 characters are truncated. Conceptually, this is a *virtual* URL or URL template that is used for selection; it is not necessarily the entire URL. The string specified here is case sensitive.

This parameter is matched against a token provided by application programs. This token might be implicitly provided by users of the Fast Response Cache Accelerator (FRCA) function, in which case the token is a web URI. It might also be explicitly provided by applications using the sendmsg() function with QoS classification ancillary data. See z/OS Communications Server: IP Programmer's Guide and Reference for more details on this support.

Tip: The specified character string can be a subset of the application-defined token. Specified URIs should begin with the first character of the path component of the URL.

For example, to select a URL of http://www.ibm.com:80/account/order.html, specify the following:

```
 ibm-ApplicationData = /account/order.html
```

Granularity can be determined when defining policy rules based on application defined data. For example, if the installation wants to assign a service level for all URLs under the *account* path, specify:

```
ibm-ApplicationData = /account
```

This specification would match all URLs beginning with /account (for example, /account/order/info.html).

Note:

1. When URIs are specified for Web Server requests, they have an affect on both static and dynamic content (assuming that the corresponding Web Server support is installed).
2. This parameter provides the ability to specify rules that match the application-defined token specified by any applications that are providing QoS application classification data. For more information, see z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference.

ApplicationPriority *n*

Specifies the QoS service level assigned for each application-specified priority and can have the following values:

0	Any application priority. This specification matches any application-specified priority value.
1	Specifies EXPIDITED priority.
2	Specifies HIGH priority.
3	Specifies MEDIUM priority.
4	Specifies LOW priority.
5	Specifies BESTEFFORT priority.

Restriction: ApplicationPriority is used to select traffic with a matching application-specified priority value. It does not assign a QoS service level to the traffic. That function is provided by the corresponding PolicyAction.

For more information about providing classification data for differentiated services policies from an application, see z/OS Communications Server: IP Programmer's Guide and Reference.

ConditionTimeRange

This field specifies an overall range of calendar dates and times over which a policy rule is valid. It is a string consisting of a start date and time, then a colon (:) followed by an end date and time. The first date indicates the beginning of the range, and the second date indicates the end of the range. Thus, the second date and time must be later than the first. Dates are expressed as substrings of the form yyyymmddhhmmss. Seconds are rounded to the nearest minute. Because all dates and times are converted internally to the Posix time format, do not specify dates and times before the start of the Posix epoch, which is January 1, 1970, 00:00:00 UTC.

For example, 20010101080000:20010131120000 is January 1, 2001, 0800 through January 31, 2001, noon.

Note:

1. The internal Posix time format is expressed in terms of seconds since the epoch, which means the time wraps sometime early in the year 2038. Therefore, do not specify dates or times later than this.
2. All dates and times refer to local time.

MonthOfYearMask

This string field specifies which months of the year the policy rule is valid. This attribute is formatted as a string containing 12 0's and 1's, where the 1's identify the months (beginning with January) in which the policy rule is valid. The value 000010010000, for example, indicates that a policy rule is valid only in the months May and August. If this attribute is omitted, then the policy assumes that it is valid for all twelve months.

DayOfMonthMask

This string field specifies which days of the month the policy rule is valid. The day of month mask can be 31 or 62 bits. The second 31 bits specify the days of the month in reverse order. Bit 32 is the last day of the month, bit 33 is the second from last day of month, and so on. This attribute is formatted as a string containing 31 or 62 0's and 1's, where the 1's identify the days of the month in which the policy rule is valid. The value 1110000000000000000000000000000, for example, indicates that a policy rule is valid only on the first three days of each month. For months with less than 31 days, the digits corresponding to the *missing* days are ignored.

The default is every day of the month.

DayOfWeekMask

A mask of seven bits representing the days in a week (Sunday through Saturday) that this policy rule is active. For example, *0111110* represents weekdays. The default is all week.

TimeOfDayRange

A series of time intervals that indicate the time of day, expressed in local time, during which this policy rule is active. Separate intervals with a comma. You can specify hours and optional minutes, separated by a colon. The values 0 and 24 both indicate midnight. Each interval consists of two values separated by a dash. If the second value is smaller than or equal to the first value, then the interval spans midnight. For example, the following statement would result in this policy rule being active from 5:30 PM until 8:30 AM:

```
TimeOfDayRange 0-8:30, 17:30-24
```

You can also configure the same time interval as follows:

```
TimeOfDayRange 17:30-8:30
```

The default is 24 hours.

PolicyActionReference

Indicates the name of a policy action from a policy action statement (for example, interactive) that this policy rule uses.

A maximum of four action references can be specified.

ForLoadDistribution

Specifies whether or not the policy rule is intended for Sysplex Distribution. Valid values are TRUE and FALSE. The default is FALSE. When TRUE is specified, the policy rule is used on sysplex distributor distributing stacks to route connection requests inbound from the network to one or more target stacks.

Table 77 provides mapping of the PolicyRule statement parameters to LDAP object classes and attributes.

Table 77. PolicyRule mapping to LDAP

PolicyRuleStatement parameter	LDAP object class	LDAP attribute
PolicyRulePriority	ibm-policyRule	ibm-policyRulePriority
PolicyActionReference	ibm-policyRule	ibm-policyRuleActionList
Not applicable	ibm-policyRule	ibm-policyRuleEnabled
Not applicable	ibm-policyRule	ibm-policyRuleConditionListType

Table 77. PolicyRule mapping to LDAP (continued)

PolicyRuleStatement parameter	LDAP object class	LDAP attribute
Not applicable	ibm-policyRule	ibm-policyRuleConditionList or ibm-policyRuleConditionListDN
Not applicable	ibm-policyRule	ibm-policyRuleValidityPeriodList
Not applicable	ibm-policyRule	ibm-policyRuleSequenceActions
Not applicable	ibm-policyRule	ibm-policyRoles
ForLoadDistribution	ibm-policyGroupLoadDistribution AuxClass	ibm-policyGroupForLoadDistribution
SourceAddressRange	ibm-hostConditionAuxClass	ibm-sourceIPAddressRange
DestinationAddress Range	ibm-hostConditionAuxClass	ibm-destinationIPAddressRange
SourcePortRange	ibm-applicationConditionAuxClass	ibm-sourcePortRange
DestinationPortRange	ibm-applicationConditionAuxClass	ibm-destinationPortRange
ProtocolNumberRange	ibm-applicationConditionAuxClass	ibm-protocolNumberRange
InboundInterface	ibm-routeConditionAuxClass	ibm-interface
OutboundInterface	ibm-routeConditionAuxClass	ibm-interface
ApplicationName	ibm-applicationConditionAuxClass	ibm-applicationName
ApplicationData	ibm-applicationConditionAuxClass	ibm-applicationData
ApplicationPriority	ibm-applicationConditionAuxClass	ibm-applicationPriority
Not applicable	ibm-idsIPAttackConditionAuxClass	ibm-idsIPOptionRange
Not applicable	ibm-idsTransportConditionAuxClass	ibm-idsLocalPortRange
Not applicable	ibm-idsTransportConditonAuxClass	ibm-idsRemotePortRange
Not applicable	ibm-idsTransportConditonAuxClass	ibm-idsProtocolRange
Not applicable	ibm-idsHostConditionAuxClass	ibm-idsLocalHostIPAddress
Not applicable	ibm-idsHostConditionAuxClass	ibm-idsRemoteHostIPAddress
ConditionTimeRange	ibm-policyTimePeriodConditionAuxClass	ibm-ptpConditionTime
MonthOfYearMask	ibm-policyTimePeriodConditionAuxClass	ibm-ptpConditionMonthOfYearMask
DayOfMonthMask	ibm-policyTimePeriodConditionAuxClass	ibm-ptpConditionDayOfMonthMask
DayOfWeekMask	ibm-policyTimePeriodConditionAuxClass	ibm-ptpConditionDayOfWeekMask
TimeOfDayRange	ibm-policyTimePeriodConditionAuxClass	ibm-ptpConditionTimeOfDayMask
Not applicable	ibm-policyTimePeriodConditionAuxClass	ibm-ptpConditionTimeZone
Not applicable	ibm-policyTimePeriodConditionAuxClass	ibm-ptpConditionLocalOrUtcTime

Also, for more information about policy schema definition files, see Chapter 23, “Intrusion detection services policy,” on page 1225.

Examples

For an example of the PolicyRule statement, see /usr/lpp/tcpip/samples/ pagent.conf.

Usage notes

If PolicyRulePriority is specified, the weight of PolicyRule is equal to the specified priority plus 100. Otherwise, the weight is determined by the number of parameters that are specified in the PolicyRule. The parameters that affect this weight are:
- ApplicationName
- ApplicationData
- ApplicationPriority
- SourceAddressRange
- DestinationAddressRange
- SourcePortRange
- DestinationPortRange
- InboundInterface
- OutboundInterface
- Direction *not* equal to BOTH
- ProtocolNumberRange

ServiceCategories statement

Use the ServiceCategories statement to specify the Type of Service that a flow of IP packets (for example, from a TCP connection, or UDP data) should receive end to end as they traverse the network. ServiceCategories can be repeated, with each having a different name so that they can be referenced later.

Restriction: This statement defines a Version 1 policy action.

Syntax

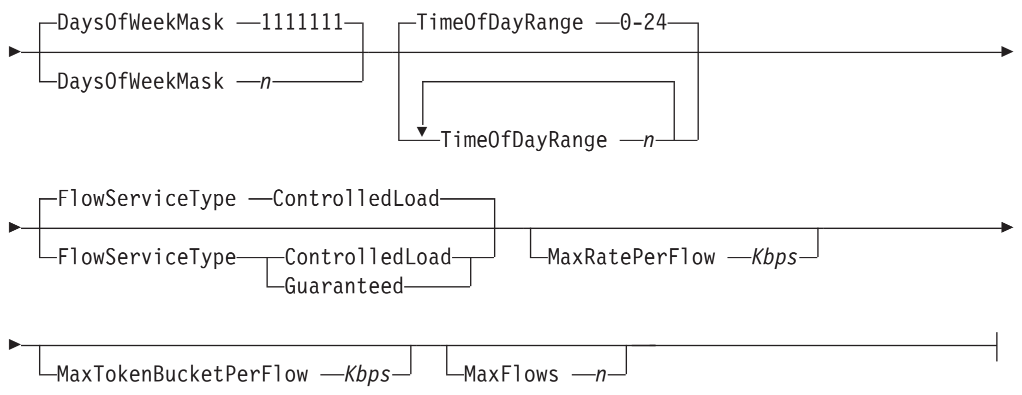
```
►►──ServiceCategories──name──┤ Place Braces and Parameters on Separate Lines ├──────►◄
```

Place Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬────────────────────────────────────┤
   ├─┤ ServiceCategories Parameters ├───┤
   └─}──────────────────────────────────┘
```

ServiceCategories Parameters:

```
├──┬──────────────────┬──┬──────────────────┬──┬────────────────────────────┬──────►
   └─MaxRate ──Kbps───┘  └─MinRate ──Kbps───┘  └─MaxDelay ──milliseconds────┘

   ┌─Interface ──All───┐  ┌─OutgoingTOS ──0──┐
►──┼───────────────────┼──┼──────────────────┼──┬─────────────────────┬──────────►
   └─Interface ──addr──┘  └─OutgoingTOS ──n──┘  └─MaxConnections ──n──┘
```



Parameters

name
A string 1 - 32 characters in length specifying the name of this service category.

MaxRate
An integer value representing the maximum rate in kilobits per second (Kbps) allowed for traffic in this service class. This attribute is valid for only or TCP. If not specified or specified as 0, there is no enforcement of the maximum rate of a connection by the local host. If a number other than 0 is specified, each TCP connection that is mapped to this ServiceCategories has its rate limited to this MaxRate. Enforcement of the MaxRate is performed by the TCP/IP stack by adjusting the TCP congestion window based on the connection round-trip time (the rate is obtained by taking the congestion window and dividing it by the round-trip time; pay attention to the units, for example, byte versus bit, second versus millisecond). Because the minimum of the congestion window is one TCP segment size, the minimum of the MaxRate that can be enforced is one TCP segment over the round-trip time. If a TCP connection has a very small round-trip delay and traverses over a very high bandwidth network (for example, Gbit Ethernet LAN), the minimum rate that this TCP connection can send (one segment per round-trip time) can be high. Therefore, users and network administrators need to know their network characteristics when setting this MaxRate; it might not be enforceable if the minimum TCP rate (for example, one segment over round-trip time) already exceeds this specified MaxRate. As noted, TCP segment size can play a role in this TCP minimum rate; for example, for a given round-trip delay, the larger the segment size the higher the minimum TCP rate. There are different factors that can affect the TCP segment size, such as the local MTU size definition, the Path MTU discovery flow (for example, this mechanism is used to discover the maximum MTU size that can be sent into the network without resulting in IP fragmentation), the receivers maximum segment size, and so on.

MinRate
An integer value representing the minimum rate or throughput (Kbps) allowed for traffic in this service class. This attribute is valid only for TCP. If not specified or specified as 0, there is no enforcement on the minimum rate of a connection by the local host. If a number other than 0 is specified, the rate for any TCP connection that is mapped to this ServiceCategories does not fall below this MinRate, unless the network is really congested and by maintaining the minimum rate the network throughput might collapse. Enforcement of the MinRate is performed by the TCP/IP stack by manipulating the congestion window over the connection round-trip time. Unlike the enforcement of MaxRate, if TCP minimum rate due to the segment size or the round-trip time,

or both, is already high, and the specified MinRate is already below this rate, it is not necessary for the TCP/IP stack to enforce the MinRate.

MaxDelay
An integer value representing the maximum delay (in milliseconds) allowed for traffic in this service class. This attribute is valid only for TCP. The TCP/IP stack does not enforce this delay.

Result: This parameter is no longer supported and is ignored.

Interface
The local IP subnet (for example, HOME statements) for which this service category applies. The default is all interfaces.

OutgoingTOS
Eight bits, left-aligned, representing the ToS value of outbound traffic belonging to this service class. The default is 0.

MaxConnections
An integer value representing the maximum number of end to end connections at any instant in time. This attribute is valid only with TCP. It places a limit on the number of TCP connections mapped to this ServiceCategories that can be active at a time. If there is a request for a new TCP connection that maps to this ServiceCategories and this limit is exceeded, the connection request is rejected. The default is that there is no policy limit. The MaxConnections attribute is enforced by the TCP/IP stack. If the connection request comes from a remote client, a TCP RST segment is returned to notify the client that the connection is refused. The number of rejected connections is kept and can be retrieved by the **netstat** command with **-j** option. If the connection request comes from an application in the local host (for example, using a connect socket call), a return code of permission denied is returned.

DaysOfWeekMask
A mask of seven bits representing the days in a week (Sunday through Saturday) that this service policy is active. For example, *0111110* represents weekdays. The default is all week.

TimeOfDayRange
A series of time intervals that indicate the time, expressed in local time, during which this service policy is active. Separate intervals with a comma. You can specify hours and optional minutes, separated by a colon. The values 0 and 24 both indicate midnight. Each interval consists of two values separated by a dash. If the second value is smaller than or equal to the first value, then the interval spans midnight. For example, the following statement results in this service policy being active from 5:30 PM until 8:30 AM:

```
TimeOfDayRange 0-8:30, 17:30-24
```

You can also configure the same time interval as follows:

```
TimeOfDayRange 17:30-8:30
```

The default is 24 hours.

FlowServiceType
Limits the Type of Service being requested by RSVP applications. Valid values are ControlledLoad (the default) and Guaranteed. Guaranteed service is considered to be *greater than* ControlledLoad service. If ControlledLoad service is specified, and an application requests Guaranteed, the requested service is downgraded to ControlledLoad. To allow RSVP applications to request Guaranteed service, specify Guaranteed for this parameter. All RSVP parameters, FlowServiceType, MaxRatePerFlow, MaxTokenBucketPerFlow, and

MaxFlows are enforced by the RSVP daemon application and not by the TCP/IP stack. The TCP/IP stack, however, keeps traffic statistics of RSVP policies, which can be retrieved by using **netstat** command with the option **-j**.

MaxRatePerFlow

Specifies the maximum rate in kilobits per second for RSVP flows. RSVP reservations are based on a traffic specification (Tspec) from the sending application. The Tspec consists of the following values:

- *r* is the token bucket rate in bytes per second.
- *b* is the token bucket depth in bytes.
- *p* is the peak rate in bytes per second.
- *m* is the minimum packet size in bytes.
- *M* is the maximum packet size (MTU) in bytes.

Use this parameter to limit the *r* value of the Tspec. If an RSVP sender application requests a Tspec *r* value larger than this parameter, the request is downgraded to this parameter value.

RSVP receiving applications also specify a resource specification (Rspec) when using Guaranteed service, as part of the reservation request. The Rspec consists of the following values:

- *R* is the rate in bytes per second.
- *S* is the slack term in microseconds.

This parameter is also used to limit the *R* value of the Rspec for reservation requests from RSVP receiver applications using Guaranteed service.

This parameter is specified in kilobits per second, while the Tspec and Rspec use bytes per second. To arrive at a compatible specification, multiply the desired Tspec or Rspec value by 8, then divide by 1 000. For example, to specify a Tspec r value of 500 000 bytes per second, specify a MaxRatePerFlow value of 4 000 (500 000 * 8 / 1 000 = 4 000).

The default for this parameter is a system defined maximum.

MaxTokenBucketPerFlow

Specifies the maximum token bucket size in kilobits per second for RSVP flows. RSVP reservations are based on a traffic specification (Tspec) from the sending application. The Tspec consists of the following values:

- *r* is the token bucket rate in bytes per second.
- *b* is the token bucket depth in bytes.
- *p* is the peak rate in bytes per second.
- *m* is the minimum packet size in bytes.
- *M* is the maximum packet size (MTU) in bytes.

This parameter is used to limit the *b* value of the Tspec. If an RSVP sender application requests a Tspec *b* value larger than this parameter, the request is downgraded to this parameter value.

This parameter is specified in kilobits, while the Tspec uses bytes. To arrive at a compatible specification, multiply the desired Tspec value by 8, then divide by 1 000. For example, to specify a Tspec b value of 75 000 bytes, specify a MaxTokenBucketPerFlow value of 600 (75 000 * 8 / 1000 = 600).

The default for this parameter is a system defined maximum.

MaxFlows
: Specifies the maximum number of reserved flows allowed for RSVP applications. The default is no limit on the number of reserved flows.

Examples

Following is an example of the ServiceCategories Version 1 Action statement.

```
ServiceCategories V1Action
{
PolicyScope Both
MaxRate 10000
MinRate 2000
MaxTokenBucket 5000
Interface 9.67.116.98
OutgoingTOS 11100000
MaxDelay 50
MaxConnections 100
DaysOfWeekMask 1111111
TimeOfDayRange 08:00-13:45,13:50-24:00
FlowServiceType Guaranteed
MaxRatePerFlow 440# 55000 bytes/second
MaxTokenBucketPerFlow 48 # 6000 bytes
MaxFlows 10
}
```

Figure 33. Example of the ServiceCategories Version 1 Action statement

ServicePolicyRules statement

Use the ServicePolicyRules statement to specify characteristics of IP packets that are used to map to a corresponding service category; it defines a set of IP datagrams that should receive a particular service.

Restriction: This statement defines a Version 1 Service Policy Rule.

Syntax

```
►►──ServicePolicyRules──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬─────────────────────────────────────────────┤
   ├─┤ ServicePolicyRules Parameters ├─┤
   └─}──────────────────────────────┘
```

ServicePolicyRules Parameters:

```
   ┌─PolicyScope ──DataTraffic──┐  ┌─Direction ──Outgoing──┐
├──┼────────────────────────────┼──┼───────────────────────┼──────────────────►
   └─PolicyScope──┬─DataTraffic─┘  └─Direction──┬─Incoming─┘
                  ├─RSVP────────┤               ├─Outgoing─┤
                  └─Both────────┘               └─Both─────┘
   ┌─Permission ──Allowed──────┐  ┌─ProtocolNumber ──All─┐  ┌─Interface ──All──┐
►──┼───────────────────────────┼──┼──────────────────────┼──┼──────────────────┼──────────►
   └─Permission──┬─Allowed─┬───┘  └─ProtocolNumber ──n───┘  └─Interface ──addr─┘
                 └─Blocked─┘

   ┌─SourceAddressRange ──All───────┐  ┌─DestinationAddressRange ──All───────┐
►──┼────────────────────────────────┼──┼─────────────────────────────────────┼──────────►
   └─SourceAddressRange ──addr addr─┘  └─DestinationAddressRange ──addr addr─┘

   ┌─SourcePortRange ──All─┐  ┌─DestinationPortRange ──All─┐
►──┼───────────────────────┼──┼────────────────────────────┼─────────────────────────────►
   └─SourcePortRange ──n n─┘  └─DestinationPortRange ──n n─┘

   ┌─DaysOfWeekMask ──1111111─┐  ┌─TimeOfDayRange ──0-24─┐
►──┼──────────────────────────┼──┼───────────────────────┼───────────────────────────────►
   └─DaysOfWeekMask ──n───────┘  └─TimeOfDayRange ──n────┘

►──┬──────────────────────────────┬──────────────────────────────────────────────────────┤
   │  ┌─────────────────────────┐ │
   │  ▼                         │ │
   └───ServiceReference ──name──┴─┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this policy rule.

PolicyScope
: Indicates to what traffic this policy rule applies. Valid values are *DataTraffic*, *RSVP*, and *Both*. The default is *DataTraffic*. When RSVP (Resource reSerVation Protocol, a network protocol running on top of IP) is specified, this policy only applies to data that are specifically reserved by using RSVP. When *DataTraffic* is specified, the policy applies to all other non-RSVP data.

Direction
: Indicates the direction of traffic for which this policy rule applies. Valid values are *Incoming*, *Outgoing*, and *Both*. The default is *Outgoing*.

  Restriction: Policies are applied to TCP on a connection basis, whereas they are applied to UDP/RAW on a per-packet basis. Therefore, the Direction attribute is also mapped accordingly. More specifically, if a policy is defined for TCP, the Direction attribute applies to the direction of the connection (inbound if the local 390 host is to receive the connection request, such as incoming TCP SYN segments). If a policy is defined for UDP/RAW, Direction applies to individual packets.

Permission
: Indicates whether packets belonging to this policy rule should be discarded or allowed to proceed. Valid values are *Allowed* and *Blocked*. The default is *Allowed*.

ProtocolNumber
: This is a 1-byte field in the IP header to identify the protocol running on top of IP. Common protocols are UDP and TCP. For UDP, TCP, and RAW, this field can be specified with these names. For others, a number has to be specified (for example, 1 for ping). The default is all protocols.

Interface
The local IP subnet for which this policy rule applies. The default is all interfaces.

SourceAddressRange
The local IP address range. This field consists of two addresses, separated by a space, where the first address is less than or equal to the second address. The default is 0, which is all inclusive.

SourceAddressRange is the address range of addresses that are local to the 390 host (for example, defined by way of HOME statements in the TCP/IP configuration).

Rules:
- Include a blank or a dash (-) as a delimiter.
- If the IP address is IPv6, it cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these two types, an error message is logged.

DestinationAddressRange
The remote IP address range. This field consists of two addresses, separated by a space, where the first address is less than or equal to the second address. The default is 0, which is all inclusive.

DestinationAddressRange is the address range of the remote hosts that are communicating with the local 390 host.

Rules:
- Include a blank or a dash (-) as a delimiter.
- If the IP address is IPv6, it cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these two types, an error message is logged.

SourcePortRange
The local port range. This field consists of two port numbers, separated by a space, where the first port number is less than or equal to the second port number. The default is 0, which is all inclusive.

SourcePortRange contains the port range of the remote hosts that are communicating with the local 390 host.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

DestinationPortRange
The remote port range. This field consists of two port numbers, separated by a space, where the first port number is less than or equal to the second port number. The default is 0, which is all inclusive.

DestinationPortRange contains the address range of the remote hosts that are communicating with the local 390 host.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

DaysOfWeekMask
A mask of seven bits representing the days in a week (Sunday through Saturday) that this policy rule is active. For example, *0111110* represents weekdays. The default is all week.

TimeOfDayRange
A series of time intervals that indicate the time, expressed in local time, during

which this policy rule is active. Separate intervals with a comma. You can specify hours and optional minutes, separated by a colon. The values 0 and 24 both indicate midnight. Each interval consists of two values separated by a dash. If the second value is smaller than or equal to the first value, then the interval spans midnight. For example, the following statement results in this policy being active from 5:30 PM until 8:30 AM:

```
TimeOfDayRange 0-8:30, 17:30-24
```

You can also configure the same time interval as follows:

```
TimeOfDayRange 17:30-8:30
```

The default is 24 hours.

ServiceReference
Indicates the name of a service category from a service category statement (for example, interactive) that this policy rule uses. One or more service category names can be specified to associate this policy rule with different interfaces or different service policies depending, for example, on the time when each of those service policies are active.

Examples

Following is an example of the ServicePolicyRules Version 1 statement.

```
ServicePolicyRules V1Rule
{
PolicyScope Both
Direction Both
Permission Allowed
ProtocolNumber TCP
Interface 9.67.116.98
SourceAddressRange 9.67.100.7.9.67.100.11
DestinationPortRange 100-5000
DaysOfWeekMask 1111111
TimeOfDayRange 08:00-23:00
ServiceReference V1Action
}
```

Figure 34. Example of the ServicePolicyRules Version 1 statement

Usage notes

The weight of ServicePolicyRules is determined by the number of parameters that are specified in the ServicePolicyRules. The parameters that affect this weight are:

- SourceAddressRange
- DestinationAddressRange
- SourcePortRange
- DestinationPortRange
- Interface
- ProtocolNumber
- Direction *not* equal to BOTH
- PolicyScope *not* equal to BOTH

Reusable policy statements

This topic contains information about the following reusable policy statements:
- "IpAddr statement"
- "IpAddrGroup statement" on page 1192
- "IpAddrSet statement" on page 1193
- "IpOptionGroup statement" on page 1195
- "IpOptionRange statement" on page 1195
- "IpProtocolGroup statement" on page 1196
- "IpProtocolRange statement" on page 1197
- "IpTimeCondition statement" on page 1198
- "Ipv6NextHdrGroup statement" on page 1200
- "Ipv6NextHdrRange statement" on page 1200
- "PortGroup statement" on page 1201
- "PortRange statement" on page 1202
- "TrafficDescriptor statement" on page 1203
- "TrafficDescriptorGroup statement" on page 1205

IpAddr statement

Use the IpAddr statement to encapsulate a single IP address specification. It can be referenced from any statement that requires a single address specification. It can also be referenced from an IpAddrGroup statement.

Syntax

```
►►──IpAddr──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
            └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────┬──────────────────────────────────────────────────────────┤
   ├─Addr──ipaddress─┤
   └─}───────────────┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IpAddr statement.

  Rule: If this IpAddr statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IpAddr statement, a nonpersistent system name is created.

Addr
: A single IP address.

  Rules for AT-TLS policies:

- If the IP address is IPv6, it cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these two types, an error message is logged.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Rules for IPSec policies:

- IPv4-mapped IPv6 addresses and IPv6 addresses with the reserved prefix ::/96 are valid only for IP filter rules and for the Identity parameter on local and remote security end points. If the IPv6 address is one of these types for any other IPSec policies, an error message is logged.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Rules for IDS policies:

- If the IP address is an IPv6 address, it cannot be an IPv4-mapped IPv6 address (in hexadecimal or dotted decimal format) or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these types, an error message is logged.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Rules for Routing policies:

- If the IP address is an IPv6 address, it cannot be an IPv4-mapped address in hexadecimal or dotted decimal format or an IP address with the reserved prefix ::/96. If the IPv6 address is one of these types, then an error message is logged.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Restriction:

- This statement is not available for use with QoS policies.

IpAddrGroup statement

Use the IpAddrGroup statement to define an IP address group. An IpAddrGroup statement identifies a set of IP specifications that make up the IP address group.

Syntax

```
►►──IpAddrGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├───┬─{──────────────────────┬───────────────────────────────────────────┤
    │  ┌───────────────────┐ │
    │  ▼                   │ │
    ├────┬─IpAddr────────┬─┴─┤
    │    ├─IpAddrRef name┤   │
    │    ├─IpAddrSet─────┤   │
    │    └─IpAddrSetRef name┘│
    └─}──────────────────────┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IP address group.

IpAddr
: An inline specification of an IpAddr statement to be included in this group.

IpAddrRef
: The name of a globally defined IpAddr statement.

IpAddrSet
: An inline specification of an IpAddrSet statement to be included in this group.

IpAddrSetRef
: The name of a globally defined IpAddrSet statement.

Result for AT-TLS policies:
: When an IpAddrGroup statement contains non-continuous ranges of IP addresses, Policy Agent cannot merge these conditions into a single condition. The group's ranges are displayed by *pasearch*, as configured, with the summary condition equal to the lowest *from* value in the group to the highest *to* value in the group. If an IP address of 0.0.0.0 exists in an IpAddrGroup statement, the summary condition for this attribute is set to All. When an IpAddrGroup statement contains a mixture of IPv4 and IPv6 addresses, a summary condition cannot be created. The group's ranges are displayed by *pasearch*, as configured, with a summary condition for this attribute of All.

Result for Routing policies:
: When an IpAddrGroup statement contains non-continuous ranges of IP addresses, Policy Agent cannot merge these conditions into a single condition. The ranges of the group are displayed by *pasearch*, as configured, with the summary condition equal to the lowest *from* value in the group to the highest *to* value in the group. If an IP address of 0.0.0.0 exists in an IpAddrGroup statement, the summary condition for this attribute is set to All. When an IpAddrGroup statement contains a mixture of IPv4 and IPv6 addresses, a summary condition cannot be created. The ranges of the group are displayed by *pasearch*, as configured, with a summary condition for this attribute of All.

Rule: For IPSec, all addresses defined within this address group must be in the same address family (IPv4 or IPv6).

IpAddrSet statement

Use the IpAddrSet statement to encapsulate either a prefix or range of IP address specifications. It can be referenced from any statement that allows for a set specification of IP addresses.

Syntax

```
►►──IpAddrSet──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
               └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────┬───────────────────────────────────────────┤
   │ ┌─Prefix ipaddress/prefixLength─┐│
   │ └─Range ipaddress-ipaddress─────┘│
   └─}────────────────────────────────┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IpAddrSet statement.

  Rule: If this IpAddrSet statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IpAddrSet statement, a nonpersistent system name is created.

Prefix
: A prefix IP address specification.

  The *prefixLength* value is the number of unmasked leading bits in the *ipaddress* value. The *prefixLength* value can be in the range 0 - 32 for IPv4 addresses and from 0 - 128 for IPv6 addresses. A packet matches this condition if its unmasked bits are identical to the unmasked bits defined.

Range
: A range of IP addresses.

Rules for AT-TLS policies:

- If the IP address is an IPv6 address, it cannot be an IPv4-mapped IPv6 address in hexadecimal or dotted decimal format or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these types, an error message is logged.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Rules for IPSec policies:

- IPv4-mapped IPv6 addresses and IPv6 addresses with the reserved prefix ::/96 are valid only for IP filter rules and for the Identity parameter on local and remote security end points. If the IPv6 address is one of these types for any other IPSec policies, an error message is logged.
- IPv6 policy is installed, but is not enforceable in a stack that is not IPv6 enabled.

Rules for IDS policies:

- If the IP address is an IPv6 address, it cannot be an IPv4-mapped IPv6 address in hexadecimal or dotted decimal format or an IPv6 address with the reserved prefix ::/96. If the IPv6 address is one of these types, an error message is logged.
- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Rules for Routing policies:

- If the IP address is an IPv6 address, it cannot be an IPv4-mapped address in hexadecimal or dotted decimal format or an IP address with the reserved prefix ::/96. If the IPv6 address is one of these types, then an error message is logged.

- IPv6 policy is installed but is not enforceable in a stack that is not IPv6 enabled.

Restriction:
- This statement is not available for use with QoS policies.

IpOptionGroup statement

Use the IpOptionGroup statement to define an IP option group. An IpOptionGroup statement identifies a set of IP option specifications that make up the IP option group.

Restriction: This statement is available for use only with IDS configuration policies.

Syntax

```
►►──IpOptionGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IpOptionGroup Parameters ├─┤
   └─}──────────────────────────┘
```

IpOptionGroup Parameters:

```
      ┌──────────────────────────┐
      ▼                          │
├────┬─IpOptionRange──────────┬─┴──────────────────────────────────────────────┤
     └─IpOptionRangeRef name──┘
```

Parameters

name
: A string 1 - 32 characters in length for the name of this IpOptionGroup.

`IpOptionRange`
: An inline specification of an IpOptionRange statement to be included in this group.

`IpOptionRangeRef`
: The name of a globally defined IpOptionRange statement.

IpOptionRange statement

Use the IpOptionRange statement to encapsulate a single IP option or range of IP options. It can be referenced from any statement that allows for a set specification of IP options.

Restriction: This statement is available for use only with IDS configuration policies.

Syntax

```
►►──IpOptionRange──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                   └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IpOptionRange Parameters ├──┤
   └─}─────────────────────────────┘
```

IpOptionRange Parameters:

```
├──IpOption──┬─n───┬────────────────────────────────────────────────────────────┤
             └─n m─┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IPOptionRange statement.

  Rule: If this IPOptionRange statement is not specified inline within another statement, *name* must be provided. If a name is not specified for an inline IPOptionRange statement, a nonpersistent system name is created.

IpOption
: A single IP option or range of options.

  Valid values for *n* are 1 - 255. While there are 255 possible valid IP options, only a few are in common usage today. If an *m* value is specified, it must be greater than or equal to *n* and less than 256.

  Restriction: For IDS policy attack types RESTRICTED_IPV6_DST_OPTIONS and RESTRICTED_IPV6_HOP_OPTIONS, you cannot restrict options 0 (Pad1) or 1 (PadN). They are always allowed.

IpProtocolGroup statement

Use the IpProtocolGroup statement to define a protocol group. An IpProtocolGroup statement identifies a set of protocol specifications that make up the protocol group.

Restriction: This statement is available for use only with IDS configuration policies.

Syntax

```
►►──IpProtocolGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IpProtocolGroup Parameters ├─┤
   └─}─────────────────────────────┘
```

IpProtocolGroup Parameters:

```
├──┬─IpProtocolRange──────────┬────────────────────────────────────────────────────┤
   └─IpProtocolRangeRef name──┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this IpProtocolGroup.

IpProtocolRange
: An inline specification of an IpProtocolRange statement to be included in this group.

IpProtocolRangeRef
: The name of a globally defined IpProtocolRange statement.

IpProtocolRange statement

Use the IpProtocolRange statement to encapsulate a single protocol or range of protocols. This statement can be referenced from any statement that allows for a set specification of protocols.

Restriction: This statement is available for use only with IDS configuration policies.

Syntax

```
►►──IpProtocolRange──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                     └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ IpProtocolRange Parameters ├─┤
   └─}─────────────────────────────┘
```

IpProtocolRange Parameters:

```
├──IpProtocol──┬─n─────┬───────────────────────────────────────────────────────────┤
               └─n  m──┘
```

Parameters

name
: A string 1 -32 characters in length specifying the name of this IpProtocolRange statement.

Rule: If this IpProtocolRange statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IpProtocolRange statement, a nonpersistent system name is created.

`IpProtocol`
A single protocol or range of protocols.

Valid values for *n* are in the range 0 - 255. A protocol range consists of one or more consecutive protocol numbers. If an *m* value is specified, it must be greater than or equal to *n* and less than 256.

Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

IpTimeCondition statement

Use the IpTimeCondition statement to define when the associated rule or action is in effect.

Syntax

```
►►──IpTimeCondition──┬────────┬──┤ Put Braces and Parameters on Separate Lines ├──────►◄
                     └─name───┘
```

Put Braces and Parameters on Separate Lines:

```
├───┬─{──────────────────────────────┬──────────────────────────────────────────────────┤
    ├─┤ IpTimeCondition Parameters ├─┤
    └─}──────────────────────────────┘
```

IpTimeCondition Parameters:

```
                                        ┌─MonthOfYearMask 111111111111─┐
├──┬─────────────────────────────┬──────┼──────────────────────────────┼────────────────►
   └─ConditionTimeRange range────┘      └─MonthOfYearMask 12 n's───────┘

   ┌─DayOfMonthMask 31 1's─┐   ┌─TimeofDayRange 0-24─┐
►──┼───────────────────────┼───┼─────────────────────┼──────────────────────────────────►
   ├─DayOfMonthMask 31 n's─┤   └─TimeofDayRange n-m──┘
   └─DayOfMonthMask 62 n's─┘

   ┌─DayOfWeekMask 1111111─┐
►──┼───────────────────────┼────────────────────────────────────────────────────────────┤
   └─DayOfWeekMask 7 n's───┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this IpTimeCondition statement.

Rule: If this IpTimeCondition statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline IpTimeCondition, a nonpersistent system name is created.

`ConditionTimeRange`
This field specifies an overall range of calendar dates and times over which a policy rule or action is active. It is a string consisting of a start date and time,

then a colon (:) followed by an end date and time. The first date indicates the beginning of the range, and the second date indicates the end of the range. Thus, the second date and time must be later than the first. Dates are expressed as substrings of the form yyyymmddhhmmss. Seconds are rounded to the nearest minute. Because all dates and times are converted internally to the Posix time format, do not specify dates and times before the start of the Posix epoch, which is January 1, 1970, 00:00:00 UTC.

For example, 20010101080000:20010131120000 is January 1, 2001, 0800 through January 31, 2001, noon.

Tips:

- The internal Posix time format is expressed in terms of seconds since the epoch, which means the time wraps sometime early in the year 2038. Therefore, do not specify dates or times later than this.
- All dates and times refer to local time.

MonthOfYearMask

This string field specifies which months of the year the policy rule or action is valid. This attribute is formatted as a string containing 12 0's and 1's, where the 1's identify the months (beginning with January) in which the policy rule or action is valid. The value 000010010000, for example, indicates that a policy rule or action is valid only in the months May and August. The default is that the policy assumes that it is valid for all twelve months.

DayOfMonthMask

This string field specifies which days of the month the policy rule or action is valid. The day of month mask can be 31 or 62 bits. The second 31 bits specify the days of the month in reverse order. Bit 32 is the last day of the month, bit 33 is the second from last day of month, and so on. This attribute is formatted as a string containing 31 or 62 0's and 1's, where the 1's identify the days of the month in which the policy rule or action is valid. The value 1110000000000000000000000000000, for example, indicates that a policy rule or action is valid only on the first three days of each month. For months with less than 31 days, the digits corresponding to the missing days are ignored.

The default is every day of the month.

DayOfWeekMask

A mask of seven bits representing the days in a week (Sunday through Saturday) that this policy rule or action is active. For example, 0111110 represents weekdays. The default is every day of the week.

TimeOfDayRange

A time interval that indicates the time of day, expressed in local time, during which this policy rule or action is active. You can specify hours and optional minutes, separated by a colon. The values 0 and 24 both indicate midnight. The interval consists of two values separated by a dash. If the second value is smaller than or equal to the first value, then the interval spans midnight. For example, the following statement results in this policy rule or action being active from midnight until 8:30 AM:

```
TimeOfDayRange 0-8:30
```

The following statement results in this policy rule or action being active from 5:30 PM until 8:30 AM:

```
TimeOfDayRange 17:30-8:30
```

Ipv6NextHdrGroup statement

Use the Ipv6NextHdrGroup statement to define an IPv6 next-header group. An Ipv6NextHdrGroup statement identifies a set of IPv6 next-header specifications that make up the IPv6 next-header group.

Restrictions:

- This statement is available for use only with IDS configuration policies.
- This statement is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

Syntax

```
►►──Ipv6NextHdrGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬──────────────────────────────────────────────┤
   ├─┤ Ipv6NextHdrGroup Parameters ├─┤
   └─}──────────────────────────┘
```

Ipv6NextHdrGroup Parameters:

```
      ┌─────────────────────────────┐
      ▼                             │
├─────┬─Ipv6NextHdrRange────────────┬┴─────────────────────────────────────────┤
      └─Ipv6NextHdrRangeRef name────┘
```

Parameters

name
: A string 1 - 32 characters in length that specifies the name of this Ipv6NextHdrGroup statement.

Ipv6NextHdrRange
: An inline specification of an Ipv6NextHdrRange statement to be included in this group.

Ipv6NextHdrRangeRef
: The name of a globally defined Ipv6NextHdrRange statement.

Ipv6NextHdrRange statement

Use the Ipv6NextHdrRange statement to encapsulate a single IPv6 next-header value or a range of IPv6 next-header values. This statement can be referenced from any statement that allows for a set specification of IPv6 next-header values.

Restrictions:

- This statement is available for use only with IDS configuration policies.
- This statement is valid only for V1R13 and later releases. See "General syntax rules for Policy Agent" on page 933 for details.

Syntax

```
►►──Ipv6NextHdrRange──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
                      └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────────────┬──────────────────────────────────────────────────┤
   ├─┤ Ipv6NextHdrRange Parameters ├─────┤
   └─}──────────────────────────────────┘
```

Ipv6NextHdrRange Parameters:

```
├──Ipv6NextHdr──┬─n───┬────────────────────────────────────────────────────────────────────┤
                └─n m─┘
```

Parameters

name
: A string 1 - 32 characters in length that specifies the name of this Ipv6NextHdrRange statement.

  Rule: If you do not specify this Ipv6NextHdrRange statement inline within another statement, you must provide a *name*. If you do not specify a name for an inline Ipv6NextHdrRange statement, a nonpersistent system name is created.

Ipv6NextHdr
: A single IPv6 next-header value or a range of IPv6 next-header values. The value in the next-header field of an IPv6 header or extension header identifies the next-header in the packet, either an upper layer protocol header (such as a TCP or UDP header) or an extension header (such as a fragmentation or routing header).

  Valid values for *n* are in the range 0 - 255. If you specify an *m* value, it must be greater than or equal to *n* and less than 256.

  Rule: You must include a blank, a colon (:), or a dash (-) as a delimiter.

PortGroup statement

Use the PortGroup statement to define a port group. A PortGroup statement identifies a set of port specifications that make up the port group.

Syntax

```
►►──PortGroup──name──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
         ┌──────────────────────────┐
         ▼                          │
├──{─────┬─PortRange──────────┬─────┴──}──────────────────────────────────────┤
         └─PortRangeRef name──┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this Port group.

PortRange
An inline specification of a PortRange statement to be included in this group.

PortRangeRef
The name of a globally defined PortRange.

Restriction: This statement is available for use only with IDS configuration and AT-TLS policies.

Tip: When a PortGroup contains non-continuous ranges of port numbers, Policy Agent cannot merge these conditions into a single condition. The group's ranges are displayed by pasearch, as configured, with the summary condition for each of these respective attributes equal to the lowest *from* value in the group to the highest *to* value in the group. If a Port of value 0 exists in a PortGroup, the summary condition for this attribute is set to the range 0 - 0.

PortRange statement

Use the PortRange statement to encapsulate a single port or range of ports. It can be referenced from any statement that allows for a set specification of ports.

Syntax

```
►►──PortRange──┬──────┬──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
               └─name─┘
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{────────────────┬────────────────────────────────────────────────────────┤
   ├─Port──┬─n───┬────┤
   │       └─n m─┘    │
   └─}────────────────┘
```

Parameters

name
A string 1 - 32 characters in length specifying the name of this PortRange statement.

Rule: If this PortRange statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline PortRange statement, a nonpersistent system name is created.

Port
A single port or range of ports.

Valid values for *n* are in the range 0 - 65 535. If 0 is specified for *n*, then any port can be used. If *n* is specified as the beginning value for a range, then 0 is not a valid value.

If an *m* value is specified, it must be greater than or equal to *n* and less than 65 536.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

Restrictions:

- For IDSAttackCondition the only valid port values for *n* are 1 - 65 535.
- PortRange is available for use only with IDS configuration and AT-TLS policies.

TrafficDescriptor statement

Use the TrafficDescriptor statement to describe IP traffic in terms of one or more of the following characteristics: IP protocol, source and destination port values, job name, NetAccess security zone, and multilevel-security (MLS) label.

Restriction: The TrafficDescriptor statement is available for use only with Routing policies.

Syntax

```
►►──TrafficDescriptor──┬──────┬──────────────────────────────────────►
                       └─name─┘

►──┤ Put Braces and Parameters on Separate Lines ├──────────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{───────────────────────────────┬────────────────────────────────┤
   ├─┤ TrafficDescriptor Parameters ├─┤
   └─}───────────────────────────────┘
```

TrafficDescriptor Parameters:

```
   ┌─Protocol All──────────┐  ┌─SourcePortRange 0─────────┐
├──┴─Protocol──┬─Tcp─┬─────┴──┴─SourcePortRange──┬─n───┬──┴──────────►
               ├─6───┤                           └─n m─┘
               ├─Udp─┤
               ├─17──┤
               └─All─┘

   ┌─DestinationPortRange 0─────┐
►──┴─DestinationPortRange──┬─n───┬──┴──┬──────────────┬──┬───────────────────┬──►
                           └─n m─┘     └─Jobname name─┘  └─SecurityZone name─┘

►──┬────────────────────┬─────────────────────────────────────────────┤
   └─SecurityLabel name─┘
```

Parameters

name
: A string 1 - 32 characters in length specifying the name of this TrafficDescriptor statement.

  Rule: If this TrafficDescriptor statement is not specified inline within another statement, a *name* value must be provided. If a name is not specified for an inline TrafficDescriptor statement, a nonpersistent system name is created.

`Protocol`

A protocol that must be contained in an IP packet for the rule's action to be performed.

TCP or 6
Indicates TCP protocol.

UDP or 17
Indicates that the UDP protocol must be in the packet.

All Indicates that all protocols that are relevant to the policy type that references the TrafficDescriptor statement must be in the packet. This is the default value.

Rule: For the Routing policy type, the relevant protocols are TCP and UDP.

`SourcePortRange`

A source port that must be contained in a TCP or UDP packet for the rule's action to be performed.

Valid values for *n* are in the range 0 - 65 535. If 0 is specified for *n*, the rule applies to any source port. If *n* is specified as the beginning value for a range, then 0 is not a valid value.

If an *m* value is specified, it must be greater than or equal to the *n* value and less than 65 536.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

Restrictions:

- The SourcePortRange value is used only as a selector for a TCP or UDP packet. If the value TCP or UDP is specified for the Protocol parameter, the SourcePortRange parameter is further restricted to the protocol specified.
- For Routing policies, the value specified for the SourcePortRange parameter is the source port that must be contained in an outbound TCP or UDP packet.

`DestinationPortRange`

A destination port that must be contained in a TCP or UDP packet for the rule's action to be performed.

Valid values for *n* are in the range 0 - 65 535. If 0 is specified for *n*, then the rule applies to any destination port. If *n* is specified as the beginning value for a range, then 0 is not a valid value.

If an *m* value is specified, it must be greater than or equal to the *n* value and less than 65 536.

Rule: Include a blank, a colon (:), or a dash (-) as a delimiter.

Restrictions:

- The DestinationPortRange value is used only as a selector for a TCP or UDP packet. If the value TCP or UDP is specified for the Protocol parameter, the DestinationPortRange is further restricted to the protocol specified.
- For Routing policies, the value specified for the DestinationPortRange parameter is the destination port that must be contained in an outbound TCP or UDP packet.

`Jobname`

The *name* value specifies the job name of the application. The *name* value can be up to 8 characters in length. A trailing asterisk indicates a wildcard specification. The specified name is not case sensitive, and is translated to uppercase before being compared.

SecurityZone
The *name* value specifies the NetAccess security zone that an IP packet must match for the rule's action to be performed. The *name* value can be up to 8 characters in length. The specified name is not case sensitive.

For Routing policies, the *name* value specifies the NetAccess security zone that an outbound IP packet must match. The outbound packet's destination IP address is used to determine the packet's NetAccess security zone in the NetAccess table defined in the TCP/IP profile. For more information about network access control, see “NETACCESS statement” on page 229.

SecurityLabel
The *name* value specifies the MLS security label that an IP packet must match for the rule's action to be performed. The *name* value can be up to 8 characters in length. The specified name is not case sensitive.

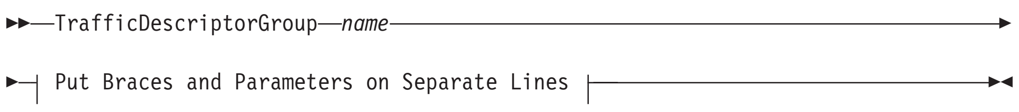
For Routing policies, the *name* value specifies the MLS security label that an outbound IP packet must match. The outbound packet's destination IP address is used to determine the packet's NetAccess security zone in the NetAccess table defined in the TCP/IP profile. The MLS security label is the label associated with the NetAccess zone. For more information, see the TCP/IP networking in a multilevel-secure environment information in z/OS Communications Server: IP Configuration Guide.

TrafficDescriptorGroup statement

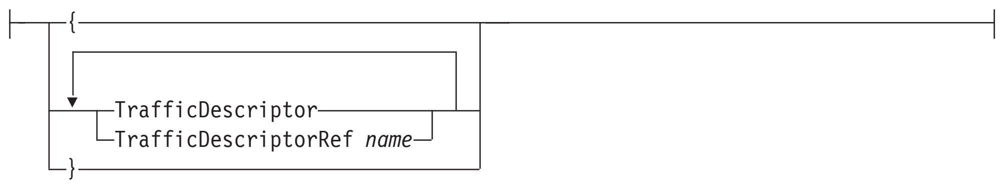
Use the TrafficDescriptorGroup statement to define a traffic descriptor group. A TrafficDescriptorGroup statement identifies a set of TrafficDescriptor statements that make up the traffic descriptor group

Restriction: The TrafficDescriptorGroup statement is available for use only with Routing policies.

Syntax



Put Braces and Parameters on Separate Lines:



Parameters

name
A string 1 - 32 characters in length specifying the name of this TrafficDescriptorGroup statement.

TrafficDescriptor
An inline specification of a TrafficDescriptor statement to be included in this group.

TrafficDescriptorRef
The name of a globally defined TrafficDescriptor statement to be included in the group.

Policy Agent search order

The search order for accessing PAGENT.CONF information is as follows. The first file found in the search order is used.

1. File or data set specified with the **-c** startup option
2. File or data set specified with the PAGENT_CONFIG_FILE environment variable
3. /etc/pagent.conf

Starting Policy Agent from the z/OS shell

The Policy Agent executable program resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure the PATH statement contains either /usr/sbin or /usr/lpp/tcpip/sbin.

The Policy Agent requires access to one or more DLLs at runtime. The LIBPATH environment variable must be set to include the /usr/lib directory, which normally includes all the required DLLs.

In order for policy time specifications to be properly acted upon, the TZ environment variable needs to be set to local time.

Set the LIBPATH and TZ environment variables as follows:

Export the LIBPATH and TZ environment variables before starting the Policy Agent. Use /etc/profile or in .profile in the HOME directory. For example, in the Eastern time zone in the United States:

```
export LIBPATH=/usr/lib
export TZ=EST5EDT4
```

See z/OS Language Environment Programming Guide for more information about specifying run time options and environment variables. Also, see z/OS UNIX System Services Command Reference for details about setting the LIBPATH and TZ environment variables.

```
►►──pagent──────────────────────────────────────────────────────────────────►

►─┬────────────────────────────────────────────────────────────────────┬──►◄
  └─-c/C ─filename──-d/D ─n──-i/I ──-t/T ─n──-l/L ─logfile──-m/M ─n──&  ┘
```

Guideline: The options can be in either upper- or lowercase (for example, C or c).

Rule: To avoid interfering with the shell session, run Policy Agent in the background. To run Policy Agent in the background, add a trailing & to the command line used to start Policy Agent.

Parameters

-c/C

The **-c**/C option allows a policy configuration file name to be specified. If it is not specified, the configuration file is located using the search order.

This value can be an z/OS UNIX or MVS data set.

The z/OS UNIX file or MVS data set is specified by the **-c** startup option. The syntax for a z/OS UNIX file is '/dir/file' and the syntax for an MVS data set is "//'MVS.DATASET.NAME'".

Tip: Note the differences in the single and double quotation marks.

-d/D

When **-d** is specified, all debug messages are logged in the Policy Agent log file. If **-d** is not used, log messages are written to the Policy Agent log file as specified by the LogLevel configuration statement. The log file should be the first place checked for error messages.

n is an integer that specifies the level of debugging. Specify a desired debug level or a combination of levels. If this start option is absent, the default level is 0. To combine debug levels, add debug level numbers. For example, to request base messages (level 1) and sysplex summary messages (Level 4), request a debug level of 5 (for example, **-d** 5).

0	None. No debug messages are logged. This is the default.
1	Base. The Policy Agent logs internal debug information. When this level is selected, the Policy Agent also uses the maximum LogLevel value, regardless of what is configured.
2	LDAP. The Policy Agent logs information about each LDAP object attribute that is processed.
4	Sysplex summary. The Policy Agent logs summary information about performance monitor QoS fraction calculations at target stacks.
8	Sysplex detail. The Policy Agent logs detailed information about performance monitor QoS fraction calculations at target stacks, and additional sysplex distributor information.
16	Memory trace. The Policy Agent logs inline details of all memory allocation and free requests. This debug level is independent of the **-m** startup option.
32	Policy install trace. The Policy Agent logs details of all policies as the policies are installed in the TCP/IP stack.
64	Lock trace. The Policy Agent logs information about locks.
128	Remote connection trace. The Policy Agent logs details about remote PAPI connections on the policy server and about connections to the policy server on the policy client.
256	Discovery connection trace. The Policy Agent logs details about requests to discover TCP/IP profile information from import requestors.

-i/I

When specified, the Policy Agent monitors its local files (all configuration files) in real time for changes. The time interval configured on the TcpImage statement is also used to monitor configuration files and the LDAP server for updates. Use of the **-i**/I option provides more timely updating of policy statements when a configuration file is changed. Change the configuration file to cause an immediate refresh of policy from the LDAP server, which causes the file to be reread. If the file is configured to read policy from the LDAP server, Policy Agent does so at that time.

Restrictions:

- Dynamic monitoring for file updates using the **-i** startup option is not supported for files configured with the DynamicConfigPolicyLoad statement.
- Dynamic monitoring for file updates using the **-i** startup option is supported only for z/OS UNIX files; MVS data sets are not monitored for changes (these files are reread at each refresh interval).

-t/T

The **-t**/T options specify whether to turn on LDAP client debugging. The following levels are supported:

0 No LDAP client debugging. This is the default.

1 This level turns on LDAP client debugging.

Tip: The destination of LDAP client debug messages is stderr.This is controlled by the LDAP client library, not the Policy Agent. This turns on the following LDAP DEBUG Options:

- LDAP_DEBUG_TRACE
- LDAP_DEBUG_PACKETS
- LDAP_DEBUG_ARGS
- LDAP_DEBUG_CONNS
- LDAP_DEBUG_BER
- LDAP_DEBUG_FILTER
- LDAP_DEBUG_MESSAGE
- LDAP_DEBUG_STATS
- LDAP_DEBUG_THREAD
- LDAP_DEBUG_PARSE
- LDAP_DEBUG_PERFORMANCE
- LDAP_DEBUG_REFERRAL
- LDAP_DEBUG_ERROR

For details about debug options, see *z/OS Security Server LDAP Client Application Development Guide and Reference*.

Restriction: If Policy Agent was started with the trace option disabled, then the output destination of stderr is closed. This option cannot later be enabled by using the MODIFY command.

-l/L logfile

The **-l**/L option can be used to specify the destination of the log output file. Either SYSLOGD or a z/OS UNIX file can be specified. If you specify SYSLOGD, you can take advantage of a centralized logging mechanism. The environment variable PAGENT_LOG_FILE also specifies the destination of the log file, using the same format as this option. The **-l**/-L option overrides the PAGENT_LOG_FILE environment variable. Another environment variable, PAGENT_LOG_FILE_CONTROL, specifies the number and size of log files (if SYSLOGD is not specified). The format is:
PAGENT_LOG_FILE_CONTROL=x,y where *x* is the log file size (kilobytes). A maximum value of 1 000 000 can be specified. *y* is the number of log files. The default is 3 log files, each 300 kilobytes in size.

The default is /tmp/pagent.log.

Result: If for some reason Policy Agent cannot read the start options, then it does not have a log file destination and Policy Agent might fail to open a z/OS UNIX log file. In these situations, Policy Agent logs error messages to the syslog daemon and exits abnormally.

If you run Policy Agent with a nonzero UID and you are using a z/OS UNIX log file, be sure to perform the following tasks:

- Specify the file permissions as either 776 or 766.
- Ensure that the syslog daemon is not configured to log the same z/OS UNIX file. The syslog daemon runs with UID 0 so Policy Agent might not be able to access its log file if syslogd creates the file before Policy Agent starts.

`-m/M n`

When specified, the Policy Agent records all memory allocation and free requests in a buffer. The number of entries in this buffer is specified on the **-m** option. The minimum value is 1 000 and the maximum value is 25 000. Values specified outside of this range are rounded up or down as needed. The number of entries in the buffer determines how many concurrent memory allocations can be recorded.

The memory request buffer can be used in two ways:

- To provide a snapshot of Policy Agent memory allocations, by using the MODIFY MEMTRC command. See z/OS Communications Server: IP System Administrator's Commands and z/OS Communications Server: IP Diagnosis Guide for more information about this command.
- To detect memory leakage by the Policy Agent. Memory leakage can only be determined when Policy Agent terminates. At the end of termination, after all memory free requests have been processed, any entries left in the memory request buffer are by definition memory leaks. If the **-m** option was specified, Policy Agent logs the contents of the memory request buffer at the end of Policy Agent termination.

If the number of entries specified on the **-m** option is too small to contain the total number of concurrent memory allocations at any point in time, Policy Agent turns off the memory trace function and stops recording in the buffer. If this occurs, the contents of the buffer are not usable, and Policy Agent logs this fact along with the high water mark number of entries at termination. Increase the number of entries the next time Policy Agent is started.

If the Policy Agent cannot successfully parse the start options, log output is written to the syslog daemon (SYSLOGD).

Starting Policy Agent as a started task

Use the S PAGENT command on an MVS console or SDSF to start Policy Agent. A sample procedure is included in member EZAPAGSP in SEZAINST. All of the information regarding default locations for the configuration and log files is the same as for starting from the z/OS shell.

The Policy Agent requires access to one or more DLLs at runtime. The LIBPATH environment variable must be set to include the /usr/lib directory, which normally includes all the required DLLs.

In order for policy time specifications to be properly acted upon, the TZ environment variable needs to be set to local time.

Set the LIBPATH and TZ environment variables as follows:

- Specify LIBPATH and TZ using the ENVAR parameter on the PARM statement in the started procedure. For example:

  ```
  // PARM=('POSIX(ON) ALL31(ON)',
  // 'ENVAR("LIBPATH=/usr/lib"',
  // '"TZ=EST5EDT4")/')
  ```

- Export the LIBPATH and TZ environment variables in a file specified with the STDENV DD statement. For example:

  ```
  //STDENV DD PATH='/etc/pagent.env',PATHOPTS=(ORDONLY)
  ```

  In the /etc/pagent.env file:

  ```
  LIBPATH=/usr/lib
  TZ=EST5EDT4
  ```

  See z/OS Language Environment Programming Guide for more information about specifying runtime options and environment variables. See z/OS UNIX System Services Command Reference for details about setting the LIBPATH and TZ environment variables.

Below is a copy of the sample procedure:

```
//PAGENT   PROC
//*
//* IBM Communications Server for z/OS
//* SMP/E distribution name: EZAPAGSP
//*
//* 5650-ZOS Copyright IBM Corp. 1998, 2013
//* Licensed Materials - Property of IBM
//* "Restricted Materials of IBM"
//* Status = CSV2R1
//*
//PAGENT   EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/'
//*
//*** Policy Agent parameters:
//*
//*   -c filename
//*
//*   Specifies the MVS data set or z/OS UNIX main configuration file.
//*   The syntax for an MVS data set is "//'MVS.DATASET.NAME'" and the
//*   syntax for a z/OS UNIX file is /dir/file.  This start option
//*   overrides the environment variable PAGENT_CONFIG_FILE if it is
//*   specified.  The default value is /etc/pagent.conf.
//*
//*   -d n
//*
//*   Specifies a debug level.  See Communications Server IP
//*   Diagnosis for a decscription of the debug levels.
//*
//*   -i
//*
//*   Specifies that Policy Agent should read configuration files
//*   immediately if they are changed.
//*
//*   -t n
//*
//*   Specifies the LDAP client trace level.  The only supported values
//*   are 1 to turn on tracing or 0 to turn off tracing.
//*
//*   -l filename | SYSLOGD
//*
//*   Specifies the destination of the Policy Agent log file.  You
//*   can specify a z/OS UNIX file name or the value SYSLOGD in
//*   uppercase.  SYSLOGD is recommended because it routes log output
//*   to the syslog daemon.  This start option overrides the environment
//*   variable PAGENT_LOG_FILE if it is specified.  The default value
//*   is /tmp/pagent.log.
//*
//*   -m n
//*
//*   Specifies that Policy Agent should enable memory tracing.  See
//*   Communications Server IP Diagnosis for more information on
```

```
//*   memory tracing.
//*
//*** Examples for specifying environment variables and parameters
//*** (parameters must extend to column 71 and be continued in
//*** column 16):
//*
//*  Example 1: Environment variables inline, MVS config data set
//*
//*      PARM=('ENVAR("LIBPATH=/usr/lib","TZ=EST5EDT")/-c //''USER.TCPP
//*            ARMS(PAGENT)'' -l SYSLOGD')
//*
//*  Example 2: Environment variables inline, UNIX config file
//*
//*      PARM=('ENVAR("LIBPATH=/usr/lib","TZ=EST5EDT")/-c /etc/pagent3.
//*            conf -l SYSLOGD')
//*
//*  Example 3: Environment variables in STDENV DD
//*
//*      PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/'
//*
//*   For this method, the STDENV DD statement below must be
//*   changed to point to a MVS data set or UNIX file containing
//*   settings for any environment variables.  For example, it should
//*   contain at least LIBPATH and TZ (unless you choose to specify TZ
//*   in a different fashion), but can contain other environment
//*   variables as in this example:
//*
//*     PAGENT_CONFIG_FILE=/etc/pagent2.conf
//*     PAGENT_LOG_FILE=SYSLOGD
//*     LIBPATH=/usr/lib
//*     TZ=EST5EDT
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//*   The use of the STDENV DD statement works well when more than
//*   one environment variable is specified, as there is a JCL limit
//*   of 100 characters on the PARM statement.
//*
//*   Note: Language Environment recommends a variable record format
//*         for the STDENV file.
//*
//*   You can also set the TZ environment variable for all applications
//*   in the CEEPRMxx PARMLIB member.  You should define the TZ
//*   environment variable for all three LE option sets (CEEDOPT,
//*   CEECOPT, and CELQDOPT).  For example:
//*
//*     CEECOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*     CEEDOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*     CELQDOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*
//*   For more information on specifying run-time options, see z/OS
//*   Language Environment Programming Guide. For details on setting
//*   the LIBPATH and TZ environment variables, see z/OS UNIX System
//*   Services Command Reference.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.PAGENT.ENV(PAGENT),DISP=SHR
//* Sample UNIX file containing environment variables:
//*STDENV   DD PATH='/etc/pagent.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.  But
//* normally, PAGENT doesn't write output to stdout or stderr.
//* Instead, output is written to the log file, which is specified
//* by the -c startup option, the PAGENT_LOG_FILE environment
//* variable, or the default of /tmp/pagent.log.  For severe
//* startup errors, such as incorrect startup options specified,
//* or being unable to open the log file, log output is instead
//* written to the syslog daemon, and help text is written to
//* stdout.
//*
```

```
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Figure 35. PAGENT sample procedure

Restriction: When you use the environment variable _CEE_ENVFILE with an MVS data set, allocate the data set with the value RECFM=V. You should not use RECFM=F because RECFM=F causes the environment variable values to be padded with blanks.

Policy Agent environment variables

Table 78 provides a list of environment variables used by Policy Agent that can be tailored to a particular installation:

Table 78. Policy Agent environment variables

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules (or a link to syntax)
PAGENT_CONFIG_FILE	PAGENT	This variable points to the location of the Policy Agent configuration file or data set.	None
PAGENT_LOG_FILE	PAGENT	Policy agent should log messages to either the syslog daemon (recommended) or a z/OS UNIX file.	Default is /tmp/pagent.log
PAGENT_LOG_FILE_CONTROL	PAGENT	Control the number and size of Policy Agent log files.	

Starting the network SLAPM2 subagent from the z/OS shell

The Network SLAPM2 Subagent executable program resides in /usr/lpp/tcpip/bin. There is also a link from /bin. Ensure that the path statement (in the profile) contains either /bin or /usr/lpp/tcpip/bin.

The Network SLAPM2 subagent requires access to one or more DLLs at runtime. The LIBPATH environment variable must be set to include the /usr/lib directory, which normally includes all the required DLLs.

Export the LIBPATH environment variable before starting the subagent. Use /etc/profile or .profile in the HOME directory. For example:

```
export LIBPATH=/usr/lib
```

For more information about specifying runtime options and environment variables, see z/OS Language Environment Programming Guide. For details about setting the LIBPATH environment variable, see z/OS UNIX System Services Command Reference.

```
                  ┌─────────────────────────┐
                  ▼                         │
►►──nslapm2──┬──────────────────────┬──┴──────────────────────────────────────►◄
             ├─-d ──n───────────────┤
             ├─-o───────────────────┤
             ├─-c ──community───────┤
             ├─-P ──port────────────┤
             ├─-p ──tcpipProcName───┤
             ├─-t ──cacheTime───────┤
             └─-?───────────────────┘
```

Parameters

-d *n*
Specifies the level of tracing to be started. The valid values for level are in the range 0 - 511. If the **-d** parameter is not specified, then a level of 3 is used, meaning all Network SLAPM2 Subagent Error, System Console and Warning Messages are traced. There are nine levels of tracing provided. Each level selected has a corresponding number. The sum of the numbers associated with each level of tracing selected is the value which should be specified as level. After the Network SLAPM2 Subagent is started, tracing options can be dynamically changed using the MVS MODIFY command. For more information about agent tracing, see z/OS Communications Server: IP Diagnosis Guide.

The numbers for the trace levels are:

0	No tracing
1	Trace Network SLAPM2 Subagent Error and System Console Messages
2	Trace Network SLAPM2 Subagent Warning Message
4	Trace Network SLAPM2 Subagent Informational Message
8	Trace Network SLAPM2 Subagent Internal statistics table
16	Trace Network SLAPM2 Subagent Internal monitor table
32	Trace Network SLAPM2 Subagent Internal traps
64	Trace Network SLAPM2 Subagent Internal monitoring
128	Trace Network SLAPM2 Subagent Internal Policy Agent API
256	Trace DPIdebug()level 2

Output from the **-d** parameter is written to syslogd or stdout depending on the **-o** parameter. The debug level can be dynamically changed using a MODIFY command.

-o Specifies that debug output is written to stdout. The default is to write to syslogd.

-c *community*
A string 1 - 32 characters in length used as the SNMP community name (or password) in establishing contact with the SNMP Agent. For nslapm2 to communicate with the z/OS CS SNMP Agent, the community name specified on the **-c** startup option must match one that is defined in a data set configured to the SNMP Agent on the **-c** parameter when the SNMP Agent is started.

For more information about how the community name is used to permit access to the SNMP agent, see Step 1: Configure the SNMP agent (OSNMPD), in z/OS Communications Server: IP Configuration Guide. The default value is public.

Tip: The community name is case sensitive.

-P *port*
A port number in the range 1 - 65 535 used in establishing communication with the SNMP Agent. For nslapm2 to communicate with the z/OS CS SNMP Agent, the port number specified must match the port number specified on the **-p** parameter when the SNMP Agent is started. The default value is 161.

-p *tcpipProcName*
The tcpipProcName is an 8-byte procedure name that is used to start TCP/IP. If this parameter is not specified, nslapm2 uses the standard resolver configuration search order to determine this parameter.

-t *cacheTime*
Amount of time in seconds to elapse before rebuilding the MIB object tables. Default value is 30 seconds.

The MinimumSamplingInterval value on the Policy Agent configuration statement PolicyPerformanceCollection is used to rebuild the MIB object tables, if the cacheTime is smaller than the MinimumSamplingInterval value.

If the MinimumSamplingInterval is updated in Pagent, then the Network SLAPM2 Subagent becomes aware of the updated time the next time it rebuilds the MIB objects.

The cache Time can be dynamically changed using a MODIFY command.

The MODIFY,QUERY command can be issued to determine the value for the MinimumSamplingInterval and this cacheTime value.

-? Display nslapm2 options help information.

Starting the network SLAPM2 subagent as a started task

Use the S NSLAPM2 command on an MVS console or SDSF to start the Network SLAPM2 subagent. A sample procedure is included in member EZAPAGSB in SEZAINST.

- Specify LIBPATH using the ENVAR parameter on the PARM statement in the started procedure. For example:

```
// PARM=('POSIX(ON) ALL31(ON)',
// 'ENVAR("LIBPATH=/usr/lib")/')
```

- Export the LIBPATH environment variable in a file specified with the STDENV DD statement. For example:

```
//STDENV DD PATH='/etc/nslapm2.env',PATHOPTS=(ORDONLY)
```

  In the /etc/nslapm2.env file:

```
LIBPATH=/usr/lib
```

For more information about specifying runtime options and environment variables, see z/OS Language Environment Programming Guide. For details about setting the LIBPATH environment variable, also see z/OS UNIX System Services Command Reference.

Following is a copy of the sample procedure:

```
//NSLAPM2 PROC
//*
//* z/OS Communications Server IP
//* SMP/E distribution name: EZAPAGSB
//*
//* 5650-ZOS Copyright IBM Corp. 2006, 2013
//* Licensed Materials - Property of IBM
//*
//NSLAPM2 EXEC PGM=NSLAPM2,REGION=0K,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/'
//*
//* Example of passing parameters to the program (parameters must
//* extend to column 71 and be continued in column 16):
//*    PARM='ENVAR("_CEE_ENVFILE_S=DD:STDENV")/-c public -P 1234 -p TCP
//*            IP'
//*
//* Provide environment variables to run with the desired stack.  As
//* an example, the data set or file specified by STDENV could
//* contain:
//*
//*   RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA)'
//*   LIBPATH=/usr/lib
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//* For information on the above environment variable, refer to the
//* IP Configuration Guide.  Other environment variables can also be
//* specified via STDENV.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.NSLAPM2.ENV(NSLAPM2),DISP=SHR
//* Sample HFS file containing environment variables:
//*STDENV   DD PATH='/etc/nslapm2.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.  But
//* normally, NSLAPM2 doesn't write output to stdout or stderr.
//* Instead, output is written to syslogd.  When the -o parameter
//* is specified, however, output is written to stdout instead of
//* syslogd.
//*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Figure 36. NSLAPM2 sample procedure

Restriction: When you use the environment variable _CEE_ENVFILE with an MVS data set, allocate the data set with the value RECFM=V. You should not use RECFM=F because RECFM=F causes the environment variable values to be padded with blanks.

Network SLAPM2 subagent environment variables

Table 79 provides a list of environment variables used by Network SLAPM2 subagent that can be tailored to a particular installation:

Table 79. Network SLAPM2 subagent environment variables

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules (or a link to syntax)
SNMP_PORT	nslapm2	Specifies the port to which a DPI subagent directs a connection query. The default is 161 (the default port on which the SNMP agent listens for queries).	None

Starting the traffic regulation manager daemon (TRMD) from the z/OS shell

TRMD is used with Traffic Regulation (TR), intrusion detection services (IDS) and IP Security to write event messages and statistics to the syslog daemon (syslogd).

The offset from Coordinated Universal Time (UTC) of the syslog time in the timestamp of TRMD messages is determined by the TZ environment variable described in z/OS UNIX System Services User's Guide.

To cause the timestamp to appear in Coordinated Universal Time (UTC), change the TZ specification in `/etc/profile` or export TZ="0" before starting TRMD.

The **-p** start option or the resolver configuration file is used to determine the stack that TRMD uses.

Syntax

```
►►──trmd──┬──────┬──┬──────────────┬──────────────────────────────────────►◄
          └─-d n─┘  └─-p stackname─┘
```

Parameters

-d *n*
Specifies that the TRMD should run in debugging mode. The following modes are supported:

1	Internal debugging messages are written.
2	Internal and API debugging messages are written.
3	Internal debugging messages and output from the ioctls issued to the stack are written.

-p *stackname*
Specifies the TCP/IP stack name that TRMD uses. If this parameter is not specified, TRMD uses the resolver configuration file to determine the stack name.

Output is written to syslogd.

Starting the traffic regulation manager daemon (TRMD) as a started task

The offset from Coordinated Universal Time (UTC) of the syslog time in the timestamp of TRMD messages is determined by the TZ environment variable described in z/OS UNIX System Services User's Guide.

To cause the timestamp to appear in coordinated universal time (UTC), specify the TZ environment variable in the start TRMD procedure. For example:

```
//   PARM=('POSIX(ON) ALL31(ON)',
//   'ENVAR("LIBPATH=/usr/lib"',
//   '"TZ=0")/-d 1')
```

Use the S TRMD command on an MVS console or SDSF to start the TRM daemon. A sample procedure is included in member EZATRMD in SEZAINST.

```
//TRMD      PROC
//*
//*   IBM Communications Server for z/OS
//*   SMP/E distribution name: EZATRMDP
//*
//*   5650-ZOS Copyright IBM Corp. 1996, 2013
//*   Licensed Materials - Property of IBM
//*   "Restricted Materials of IBM"
//*
//*   Status = CSV2R1
//*
//*   Function: Sample procedure for running the Traffic
//*             Regulator Management Daemon (TRMD)
//*
//TRMD   EXEC PGM=EZATRMD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)',
//   'ENVAR("LIBPATH=/usr/lib")/')
//*
//*** Notes:
//*
//* - TRMD can also be invoked from the Unix System Services shell
//*   as a shell command: trmd
//*
//* - The system link list concatenation must contain the TCP/IP
//*   runtime libraries and the C runtime libraries.  If they are
//*   not in the link list concatenation, this procedure will need
//*   to be changed to STEPLIB to them.
//*
//* - To pass parameters to TRMD, specify them after the final slash
//*   on the PARM statement.  For example:
//*      //   PARM=('POSIX(ON) ALL31(ON)/-d 1')
//*
//* - TRMD must find the TCP/IP job name with which it should be
//*   associated.  It uses the -p start option or the TCPIPJOBNAME value
//*   from the TCPIP.DATA file.  The TCPIP.DATA file used can be
//*   controlled by setting the RESOLVER_CONFIG environment variable.
//*   See examples below.
//*
//*** Examples for specifying environment variables and parameters
//*** (parameters must extend to column 71 and be continued in
//*** column 16):
//*
//*  Example 1: Environment variables inline, MVS resolver data set
//*
//*   PARM=('POSIX(ON) ALL31(ON)',
//*   'ENVAR("RESOLVER_CONFIG=//''SYS1.TCPPARMS(TCPDATA)''","TZ=EST5EDT
//*            ")/')
//*
//*  Example 2: Environment variables inline, UNIX resolver file
//*
//*   PARM=('POSIX(ON) ALL31(ON)',
//*   'ENVAR("RESOLVER_CONFIG=/etc/resolv.conf","TZ=EST5EDT")/')
//*
//*  Example 3: Environment variables in STDENV DD
//*
```

```
//*   PARM=('POSIX(ON) ALL31(ON)',
//*   'ENVAR("_CEE_ENVFILE_S=DD:STDENV")/')
//*
//*   For this method, the STDENV DD statement below must be
//*   changed to point to a MVS data set or UNIX file containing
//*   settings for any environment variables.  For example, it should
//*   contain at least TZ (unless you choose to specify TZ in a
//*   different fashion), but can contain other environment variables
//*   as in this example:
//*
//*     RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA)'
//*     TZ=EST5EDT
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//*   The use of the STDENV DD statement works well when more than
//*   one environment variable is specified, as there is a JCL limit
//*   of 100 characters on the PARM statement.
//*
//*   Note: Language Environment recommends a variable record format
//*         for the STDENV file.
//*
//*   You can also set the TZ environment variable for all applications
//*   in the CEEPRMxx PARMLIB member.  You should define the TZ
//*   environment variable for all three LE option sets (CEEDOPT,
//*   CEECOPT, and CELQDOPT).  For example:
//*
//*     CEECOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*     CEEDOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*     CELQDOPT(ALL31(ON), ENVAR('TZ=EST5EDT') )
//*
//*   For more information on specifying run-time options, see z/OS
//*   Language Environment Programming Guide. For details on setting
//*   the LIBPATH and TZ environment variables, see z/OS UNIX System
//*   Services Command Reference.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.TRMD.ENV(TRMD),DISP=SHR
//* Sample UNIX file containing environment variables:
//*STDENV   DD PATH='/etc/trmd.env',PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Figure 37. TRMD sample procedure

Chapter 22. RSVP Agent

Restriction: IPv6 support is not provided for RSVP agent at this time.

For related information about RSVP Agent, see the policy based networking information in z/OS Communications Server: IP Configuration Guide.

RSVP Agent configuration file

The RSVP Agent uses the following search order to locate the configuration file (highest priority is listed first):

- z/OS UNIX file or MVS data set specified by the -c startup option. The syntax for a z/OS UNIX file is '/dir/file' and the syntax for an MVS data set is "//'MVS.DATASET.NAME'".
- z/OS UNIX file or MVS data set specified with the RSVPD_CONFIG_FILE environment variable.
- /etc/rsvpd.conf z/OS UNIX file.
- '*hlq*.RSVPD.CONF' MVS data set.

Restriction: If this file is not present, RSVP is enabled on all network interfaces with default parameters.

LogLevel statement

Use the LogLevel statement to specify the level of tracing.

Syntax

```
►►──LogLevel── i──────────────────────────────────────────────────────►◄
```

Parameters

i

An integer that specifies the level of logging/tracing. The supported levels are:

- 1 - SYSERR - System error messages
- 2 - OBJERR - Object error messages
- 4 - PROTERR - Protocol error messages
- 8 - WARNING - Warning messages
- 16 - EVENT - Event messages
- 32 - ACTION - Action messages
- 64 - INFO - Informational messages
- 128 - ACNTING - Accounting messages
- 256 - TRACE - Trace messages

Usage notes

Specify a desired log level or a combination of levels. If this statement is absent, the default level is 15.

To combine log levels, add log level numbers. For example, to request SYSERR messages (level 1) and EVENT messages (level 16), you would request log level 17.

Examples

The following example turns on all trace levels for RSVP.

```
LogLevel 511
```

TcpImage statement

Use the TcpImage statement to identify the name of the stack to which the RSVP agent should establish affinity.

Rule: If the TcpImage statement is absent, the RSVP agent establishes affinity with the default stack.

Syntax

```
►►──TcpImage──name──────────────────────────────────────────────────►◄
```

Parameters

name
The name of the TCP/IP image. The name must be one to eight characters.

Examples

```
TcpImage TCPCS2
```

Interface statement

Use the Interface statement to make available to the RSVP agent one or more of the network interfaces of the local host.

Rule: If the Interface statement is absent, none of the network interfaces are available to the RSVP agent.

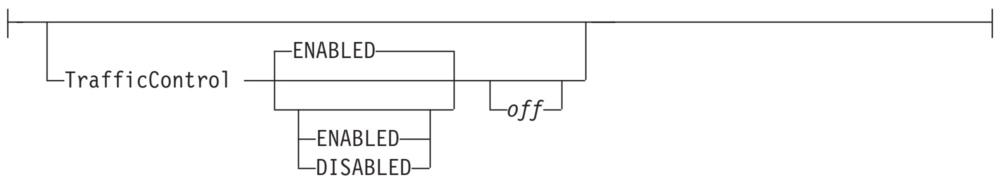
Syntax

```
                              ┌─ENABLED────────┐
►►──Interface──┬─ALL─────────┬─┼────────────────┼──┬─────┬─┤ See Note 1 ├──────►◄
               ├─OTHERS──────┤ └─┬─ENABLED──┬───┘  └─off─┘
               └─ip_address──┘   └─DISABLED─┘
```

Note 1: Place braces and parameters on separate lines:

```
├──┬─{──────────────────────────┬────────────────────────────────────────┤
   ├─┤ Interface parameters ├───┤
   └─}──────────────────────────┘
```

Interface Parameters:



Parameters

IP_address
: The IP address (dotted decimal format) of the interface. You can choose a specific interface IP address such as *all*, which means all configured interfaces (currently configured on the HOME statement or dynamically added in the future), or *others*, which means all interfaces except those previously configured.

  In the following example, all interfaces except 9.10.11.12 would be enabled.

```
Interface 9.10.11.12 Disabled
Interface others Enabled
```

Enabled
: Specifies that RSVP should use this interface.

Disabled
: Specifies that RSVP should not use this interface.

Off
: Specifies to ignore this statement.

TrafficControl
: Specifies whether or not traffic control is in effect. When traffic control is disabled, the RSVP agent does not install any filters (resource reservations). If off is specified, the traffic control specification portion of the Interface statement is ignored.

Examples

```
Interface 9.23.78.13
{
  trafficcontrol enabled
}
```

```
interface others disabled
```

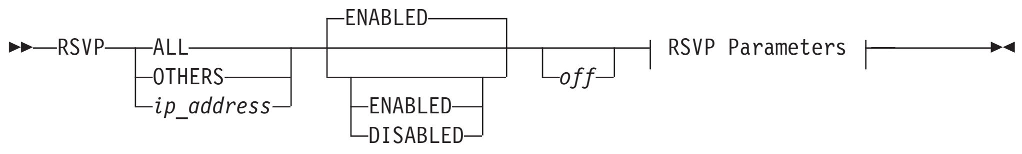
```
interface all
```

RSVP statement

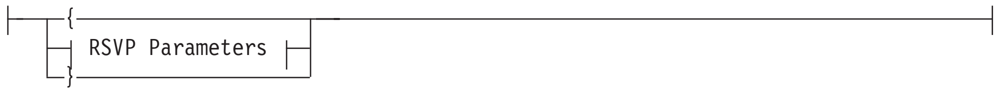
Use the RSVP statement to enable RSVP processing on one or more of the network interfaces of the local host.

Rule: If this statement is absent, RSVP processing is disabled on all network interfaces.

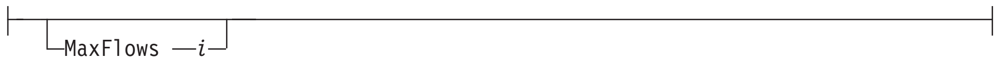
Syntax



Put Braces and Parameters on Separate Lines:



RSVP Parameters:



Parameters

`IP_address`
The IP address (dotted decimal format) of the interface. You can choose a specific interface IP address such as *all*, which means all configured interfaces (currently configured on the HOME statement or dynamically added in the future), or *others* which means all interfaces except those previously configured.

In the following example, all interfaces except 9.10.11.12 would be enabled.

```
Interface 9.10.11.12 Disabled
Interface others Enabled
```

`Enabled`
Specifies that RSVP processing should use this interface.

`Disabled`
Specifies that RSVP processing should not use this interface.

`Off`
Specifies to ignore this statement.

`MaxFlows`
Specifies the maximum number of data flows.

i An integer defining the maximum number of data flows to be allowed using this interface. The default is 32.

Examples

```
rsvp 87.13.112.6

{
maxflows 100
}

rsvp others

rsvp all
```

RSVPD.CONF search order

The search order for accessing RSVPD.CONF information is as follows. The first file found in the search order is used.

1. z/OS UNIX file or MVS data set specified by the -c startup option. The syntax for a z/OS UNIX file is '/dir/file' and the syntax for an MVS data set is "//'MVS.DATASET.NAME'".
2. z/OS UNIX file or MVS data set specified with the RSVPD_CONFIG_FILE environment variable.
3. /etc/rsvpd.conf z/OS UNIX file.
4. '*hlq*.RSVPD.CONF' MVS data set.

Restriction: If this file is not present, RSVP is enabled on all network interfaces with default parameters.

Starting RSVP from the z/OS shell

The rsvp executable program resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin.

Requirement: Make sure your path statement (in the profile) contains either /usr/sbin or /usr/lpp/tcpip/sbin.

```
►►──rsvpd──┬───────────────┬────────────────────────────────────────────────►◄
           └─-c ──filename─┘
```

-c The -c option allows an RSVP Agent configuration file to be specified. If it is not specified, the configuration file is located using the search order.

Starting RSVP as a started task

Use the S RSVPD command on an MVS console or SDSF to start RSVP. A sample procedure is shipped in member EZARSVPP in SEZAINST. All of the information regarding default locations for the configuration and log files is the same as for starting from the z/OS shell. Following is a copy of the sample procedure:

```
//RSVPD    PROC
//*
//* SecureWay Communications Server IP
//* SMP/E distribution name: EZARSVPP
//*
//* 5650-ZOS (C) Copyright IBM Corp. 1999, 2013
//* Licensed Materials - Property of IBM
//*
//RSVPD    EXEC PGM=RSVPD,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE_S=DD:STDENV")/'
//*
//* Example of passing parameters to the program (parameters must
//* extend to column 71 and be continued in column 16):
//*      PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE_S=DD:STDENV")/-c /
//*            etc/rsvpd25.conf'
//*
//* Provide environment variables to run with the desired stack and
//* configuration.  As an example, the data set or file specified by
//* STDENV could contain:
//*
//*   RESOLVER_CONFIG=//'SYS1.TCPPARMS(TCPDATA2)'
//*   RSVPD_CONFIG_FILE=/etc/rsvpd2.conf
//*   RSVPD_LOG_FILE=/tmp/rsvpd2.log
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//* For information on the above environment variables, refer to the
//* IP CONFIGURATION GUIDE.  Other environment variables can also be
//* specified via STDENV.
//*
//STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.RSVPD.ENV(RSVPD2),DISP=SHR
//* Sample HFS file containing environment variables:
//*STDENV   DD PATH='/etc/rsvpd2.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.  But
//* normally, RSVPD doesn't write output to stdout or stderr.
//* Instead, output is written to the log file, which is specified
//* by the RSVPD_LOG_FILE environment variable, and defaults to
//* /tmp/rsvpd.log.
//*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

Figure 38. RSVP sample procedure

Restriction: When you use the environment variable _CEE_ENVFILE with an MVS data set, allocate the data set with the value RECFM=V. To use a RECFM=F data set, _CEE_ENVFILE_S should be used to prevent the environment variable values from being padded with blanks.

Chapter 23. Intrusion detection services policy

This topic contains the following information about the intrusion detection services (IDS) policy:

- "IDS policies defined in IDS configuration files"
- "IDS Policies defined in LDAP"

IDS policies defined in IDS configuration files

For information about configuring the intrusion detection services (IDS) rules and actions in the IDS configuration files, see Chapter 21, "Policy Agent and policy applications," on page 931.

If you are migrating your IDS policy from Lightweight Directory Access Protocol (LDAP) to IDS configuration files, be aware that the IDS configuration file policy rule and action parameters are consistent with the IDS LDAP policy rule and action parameters, except for the following parameters:

- IDSAttackCondition
  - IfcFloodPercentage
  - IfcFloodMinDiscard
- IDSScanEventCondition
  - Sensitivity
  - IDSScanExclusion
  - IDSScanExclusionRef
- IDSScanGlobalCondition
  - FSInterval
  - FSThreshold
  - SSInterval
  - SSThreshold
- IDSTRCondition
  - TRtcpTotalConnections
  - TRudpQueueSize
  - TRtcpPercentage
  - TRtcpLimitScope

IDS Policies defined in LDAP

This topic lists the LDAP object classes and attributes used to define IDS policy objects. The default and allowable values for IDS-specific attributes are included, as well as information showing the allowable combinations of attributes in various types of IDS policies. See Appendix B, "LDAP definition files," on page 1433 for more information about object classes and their attributes. See z/OS Communications Server: IP Configuration Guide for additional guidance about defining IDS policies.

Restriction: Not all IDS policy options are available in Lightweight Directory Access Protocol (LDAP) configuration file. You cannot use LDAP to do the following tasks:

- Specify that ICMPv6 traffic should be monitored for scan events
- Exclude IPv6 addresses from the scan exclusion list
- Define rules for the following attack types:
  - DATA_HIDING
  - EE_LDLC_CHECK
  - EE_MALFORMED_PACKET
  - EE_PORT_CHECK
  - EE_XID_FLOOD
  - GLOBAL_TCP_STALL
  - OUTBOUND_RAW_IPV6
  - RESTRICTED_IPV6_DST_OPTIONS
  - RESTRICTED_IPV6_HOP_OPTIONS
  - RESTRICTED_IPV6_NEXT_HDR
  - TCP_QUEUE_SIZE
- Define TCP traffic regulation policy that specifies IPv6 addresses
- Define UDP traffic regulation policy that specifies IPv6 addresses

The following Object classes are useful in building an LDAP tree structure of policy groups of rules and policy repositories of reusable conditions and actions.
- objectclass ibm-policy
- objectclass ibm-policyGroup
- objectclass ibm-policyRepository
- objectclass ibm-policyGroupContainmentAuxClass
- objectclass ibm-policyRuleContainmentAuxClass

The following Object classes are useful in building IDS rule, condition association, rule-specific condition, reusable condition, action association, rule-specific action and reusable action objects.
- objectclass ibm-policyRule
- objectclass ibm-policyRuleConditionAssociation
- objectclass ibm-policyRuleActionAssociation
- objectclass ibm-policyInstance
- objectclass ibm-policyConditionInstance
- objectclass ibm-policyActionInstance
- objectclass ibm-policyConditionAuxClass
- objectclass ibm-policyActionAuxClass
- objectclass ibm-policyTimePeriodConditionAuxClass

The following Object classes are required for IDS-specific condition objects. These classes are not permitted in QoS specific policies.
- objectclass ibm-idsConditionAuxClass
- objectclass ibm-idsAttackConditionAuxClass
- objectclass ibm-idsIPAttackConditionAuxClass
- objectclass ibm-idsFloodAttackActionsAuxClass
- objectclass ibm-idsTrafficRegulationConditionAuxClass
- objectclass ibm-idsScanConditionAuxClass
- objectclass ibm-idsScanEventConditionAuxClass

- objectclass ibm-idsTransportConditionAuxClass
- objectclass ibm-idsHostConditionAuxClass

The following Object classes are required for IDS-specific action objects. These classes are not permitted in QoS specific policies.

- objectclass ibm-idsActionAuxClass
- objectclass ibm-idsNotificationAuxClass
- objectclass ibm-idsAttackActionsAuxClass
- objectclass ibm-idsTrafficRegulationActionAuxClass
- objectclass ibm-idsTRtcpActionAuxClass
- objectclass ibm-idsTRudpActionAuxClass
- objectclass ibm-idsScanActionAuxClass
- objectclass ibm-idsScanSensitivityActionAuxClass
- objectclass ibm-idsScanExclusionActionAuxClass

The following Object classes are not permitted in IDS specific objects either because they are only valid for Version 2 policies or because they are only permitted in QoS specific objects.

- objectclass ibm-policyCondition
- objectclass ibm-policyTimePeriodCondition
- objectclass ibm-networkingPolicyCondition
- objectclass ibm-policyAction
- objectclass ibm-serviceCategories
- objectclass ibm-networkingPolicyConditionAuxClass
- objectclass ibm-routeConditionAuxClass
- objectclass ibm-hostConditionAuxClass
- objectclass ibm-applicationConditionAuxClass
- objectclass ibm-serviceCategoriesAuxClass
- objectclass ibm-policyGroupLoadDistributionAuxClass
- objectclass SetSubnetPrioTosMask

IDS-specific condition attributes, their object class, as well as allowed and default values are listed in Table 80.

Table 80. IDS-specific condition attributes

Attribute	**Class**	**Allowed and default values**
ibm-idsConditionType	ibm-idsConditionAuxClass	• ATTACK • TR • SCAN_GLOBAL • SCAN_EVENT No default

Table 80. IDS-specific condition attributes (continued)

Attribute	Class	Allowed and default values
ibm-idsAttackType	ibm-idsAttackConditionAuxClass	• MALFORMED_ PACKET • FLOOD • OUTBOUND_RAW • PERPETUAL_ECHO • IP_FRAGMENT • RESTRICTED_IP_ OPTIONS • RESTRICTED_IP_ PROTOCOL • ICMP_REDIRECT No default
ibm-idsIPOptionRange	ibm-idsIPAttackConditionAuxClass	1 - 255 Default is 0 (all)
ibm-idsLocalPortRange	ibm-idsTransportConditionAuxClass	0–65535 Default is 0 (all)
ibm-idsRemotePortRange	ibm-idsTransportConditionAuxClass	0 - 65535 Default is 0 (all)
ibm-idsProtocolRange	ibm-idsTransportConditionAuxClass	0 - 255 Default is Protocol 0
ibm-idsLocalHostIPAddress	ibm-idsHostConditionAuxClass	Any valid IP address Default is 0 (all)
ibm-idsRemoteHostIPAddress	ibm-idsHostConditionAuxClass	Any valid IP address Default is 0 (all)

IDS-specific action attributes, their object class, and allowed and default values are shown in Table 81.

Table 81. IDS-specific action attributes

Attribute	Class	Allowed values
ibm-idsIfcFloodPercentage	ibm-idsFloodAttackActionsAuxClass	5 - 100 Default is 10.
ibm-idsIfcFloodMinDiscard	ibm-idsFloodAttackActionsAuxClass	100 - 4 294 967 295 Minimum number of discards that must occur in a one minute interval for an interface flood condition to exist. Default is 1 000.

Table 81. IDS-specific action attributes (continued)

Attribute	Class	Allowed values
ibm-idsActionType	ibm-idsActionAuxClass	• ATTACK • TR • SCAN_GLOBAL • SCAN_EVENT No default
ibm-idsNotification	ibm-idsNotificationAuxClass	• NONE • SYSLOG • SYSLOGDETAIL • CONSOLE No default
ibm-idsStatInterval	ibm-idsNotificationAuxClass	0 - 4 294 967 295 Default is 60
ibm-idsLoggingLevel	ibm-idsNotificationAuxClass	0 - 7 These values map to syslogd priority levels as follows: **0** Emerg/panic **1** Alert **2** Crit **3** Error **4** Warning **5** Notice **6** Info **7** Debug Default is 0
ibm-idsTypeActions	ibm-idsNotificationAuxClass	• STATISTICS • EXCEPTSTATS • LOG • LIMIT No default
ibm-idsTraceData	ibm-idsNotificationAuxClass	• NONE • HEADER • FULL • RECORDSIZE Default is HEADER
ibm-idsTraceRecordSize	ibm-idsNotificationAuxClass	0 - 4 294 967 295 Default is 100
ibm-idsMaxEventMessage	ibm-idsAttackActionsAuxClass	0 - 4 294 967 295 Default is 0
ibm-idsTRtcpTotalConnections	ibm-idsTRtcpActionAuxClass	0 - 65 535 Default is 65535

Table 81. IDS-specific action attributes (continued)

Attribute	Class	Allowed values
ibm-idsTRtcpPercentage	ibm-idsTRtcpActionAuxClass	0 - 100 Default is100
ibm-idsTRtcpLimitScope	ibm-idsTRtcpActionAuxClass	• PORT • PORT_INSTANCE Default is PORT_INSTANCE
ibm-idsTRudpQueueSize	ibm-idsTRudpActionAuxClass	• VERY_LONG • LONG • SHORT • VERY_SHORT Default is VERY_LONG
ibm-idsFSInterval	ibm-idsScanActionAuxClass	1 - 1440 Default is 1
ibm-idsFSThreshold	ibm-idsScanActionAuxClass	1 - 64 Default is 5
ibm-idsSSInterval	ibm-idsScanActionAuxClass	0 - 1 440 Default is 120
ibm-idsSSThreshold	ibm-idsScanActionAuxClass	0 - 64 Default is 10
ibm-idsSensitivity	ibm-idsScanSensitivityActionAuxClass	• NONE • HIGH • MEDIUM • LOW No default
ibm-idsScanExclusion	ibm-idsScanExclusionActionAuxClass	Any valid IP address, 0 - 65 535 for ports Default is 0 (none)

The tables in this topic list the combinations of attributes that are used for different types of IDS policy. Mapping conditions are the attributes used by the code when searching for rules.

Use the following guidelines for interpreting the following tables:
- Quoted strings are literal attribute values.
- X indicates *not* supported; the containing policy is not mapped.
- I indicates ignored.
- A indicates allowed.
- R indicates required.

Table 82 on page 1231 lists the IDS scan global policies.

Table 82. IDS scan global policies

Mapping conditions	
ibm-idsConditionType	"SCAN_GLOBAL"
Other Conditions	
ibm-idsAttackType	X
ibm-idsIPOptionRange	X
ibm-idsLocalPortRange	X
ibm-idsRemotePortRange	X
ibm-idsProtocolRange	X
ibm-idsLocalHostIPAddress	X
ibm-idsRemoteHostIPAddress	X
Actions	
ibm-idsActionType	"SCAN_GLOBAL" (1)
ibm-idsTypeActions	A (2)
ibm-idsNotification	A
ibm-idsLoggingLevel	A
ibm-idsStatInterval	I
ibm-idsMaxEventMessage	I
ibm-idsTraceData	A
ibm-idsTraceRecordSize	A
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	A
ibm-idsFSThreshold	A
ibm-idsSSInterval	A
ibm-idsSSThreshold	A
ibm-idsSensitivity	I
ibm-idsScanExclusion	I
ibm-idsIfcFloodPercentage	I
ibm-idsIfcFloodMinDiscard	I
Notes: 1. Additional values are allowed in same action. 2. STATISTICS, EXCEPTSTATS ignored.	

Table 83 lists the IDS scan event policies.

Table 83. IDS scan event policies (ICMP)

Mapping conditions	
ibm-idsConditionType	"SCAN_EVENT" (1)
ibm-idsProtocolRange	"1" (ICMP)
Other Conditions	

Table 83. IDS scan event policies (ICMP) (continued)

Mapping conditions	
ibm-idsAttackType	X
ibm-idsIPOptionRange	X
ibm-idsLocalPortRange	X
ibm-idsRemotePortRange	X
ibm-idsLocalHostIPAddress	X
ibm-idsRemoteHostIPAddress	X
Actions	
ibm-idsActionType	"SCAN_EVENT" (2)
ibm-idsTypeActions	I
ibm-idsNotification	I
ibm-idsLoggingLevel	I
ibm-idsStatInterval	I
ibm-idsMaxEventMessage	I
ibm-idsTraceData	I
ibm-idsTraceRecordSize	I
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	I
ibm-idsFSThreshold	I
ibm-idsSSInterval	I
ibm-idsSSThreshold	I
ibm-idsSensitivity	A
ibm-idsScanExclusion	A
ibm-idsIfcFloodPercentage	I
ibm-idsIfcFloodMinDiscard	I

Notes:

1. A SCAN EVENT rule that includes ICMP in the protocol range is not mapped for ICMP if it also includes a local host IP address or port condition.
2. Additional values are allowed in same action.

Table 84 lists more IDS scan event policies.

Table 84. IDS scan event policies (TCP and UDP)

Mapping conditions	
ibm-idsConditionType	"SCAN_EVENT" (1)
ibm-idsProtocolRange	"6" (TCP) | "17" (UDP)
ibm-idsLocalHostIPAddress	A
ibm-idsLocalPortRange	A
Other conditions	

Table 84. IDS scan event policies (TCP and UDP) (continued)

Mapping conditions	
ibm-idsAttackType	X
ibm-idsIPOptionRange	X
ibm-idsRemotePortRange	X
ibm-idsRemoteHostIPAddress	X
Actions	
ibm-idsActionType	"SCAN_EVENT" (2)
ibm-idsTypeActions	I
ibm-idsNotification	I
ibm-idsLoggingLevel	I
ibm-idsStatInterval	I
ibm-idsMaxEventMessage	I
ibm-idsTraceData	I
ibm-idsTraceRecordSize	I
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	I
ibm-idsFSThreshold	I
ibm-idsSSInterval	I
ibm-idsSSThreshold	I
ibm-idsSensitivity	A
ibm-idsScanExclusion	A
ibm-idsIfcFloodPercentage	I
ibm-idsIfcFloodMinDiscard	I
Notes: 1. A SCAN EVENT rule that includes ICMP in the protocol range is not mapped for ICMP if it also includes a local host IP address or port condition. 2. Additional values are allowed in same action.	

Table 85 lists IDS attack policies.

Table 85. IDS attack policies (FLOOD)

Mapping conditions	
ibm-idsConditionType	"ATTACK"
ibm-idsAttackType	"FLOOD"
Other conditions	
ibm-idsIPOptionRange	I
ibm-idsLocalPortRange	I
ibm-idsRemotePortRange	I
ibm-idsProtocolRange	I

Table 85. IDS attack policies (FLOOD) (continued)

Mapping conditions	
ibm-idsLocalHostIPAddress	I
ibm-idsRemoteHostIPAddress	I
Actions	
ibm-idsActionType	"ATTACK" (1)
ibm-idsTypeActions	A (2)
ibm-idsNotification	A (3)
ibm-idsLoggingLevel	A
ibm-idsStatInterval	A
ibm-idsMaxEventMessage	A
ibm-idsTraceData	A
ibm-idsTraceRecordSize	A
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	I
ibm-idsFSThreshold	I
ibm-idsSSInterval	I
ibm-idsSSThreshold	I
ibm-idsSensitivity	I
ibm-idsScanExclusion	I
ibm-idsIfcFloodPercentage	A
ibm-idsIfcFloodMinDiscard	A

Notes:
1. Additional values are allowed in same action.
2. LIMIT is ignored. Packets identified as part of a flood are always discarded.
3. SYSLOGDETAIL is equivalent to SYSLOG.

Table 86 lists the IDS attack policies (MALFORMED).

Table 86. IDS attack policies (MALFORMED)

Mapping conditions	
ibm-idsConditionType	"ATTACK"
ibm-idsAttackType	"MALFORMED_PACKET"
Other conditions	
ibm-idsIPOptionRange	I
ibm-idsLocalPortRange	I
ibm-idsRemotePortRange	I
ibm-idsProtocolRange	I
ibm-idsLocalHostIPAddress	I
ibm-idsRemoteHostIPAddress	I

Table 86. IDS attack policies (MALFORMED) (continued)

Mapping conditions	
Actions	
ibm-idsActionType	"ATTACK" (1)
ibm-idsTypeActions	A (2)
ibm-idsNotification	A (3)
ibm-idsLoggingLevel	A
ibm-idsStatInterval	A
ibm-idsMaxEventMessage	A
ibm-idsTraceData	A
ibm-idsTraceRecordSize	A
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	I
ibm-idsFSThreshold	I
ibm-idsSSInterval	I
ibm-idsSSThreshold	I
ibm-idsSensitivity	I
ibm-idsScanExclusion	I
ibm-idsIfcFloodPercentage	I
ibm-idsIfcFloodMinDiscard	I

Notes:
1. Additional values are allowed in same action.
2. LIMIT is ignored. Malformed packets are always discarded.
3. SYSLOGDETAIL is equivalent to SYSLOG.

Table 87 lists more IDS attack policies.

Table 87. IDS attack policies (FRAGMENT and REDIRECT)

Mapping conditions	
ibm-idsConditionType	"ATTACK"
ibm-idsAttackType	"IP_FRAGMENT" | "ICMP_REDIRECT"
Other conditions	
ibm-idsIPOptionRange	I
ibm-idsLocalPortRange	I
ibm-idsRemotePortRange	I
ibm-idsProtocolRange	I
ibm-idsLocalHostIPAddress	I
ibm-idsRemoteHostIPAddress	I
Actions	
ibm-idsActionType	"ATTACK" (1)

Table 87. IDS attack policies (FRAGMENT and REDIRECT) (continued)

Mapping conditions	
ibm-idsTypeActions	A
ibm-idsNotification	A (2)
ibm-idsLoggingLevel	A
ibm-idsStatInterval	A
ibm-idsMaxEventMessage	A
ibm-idsTraceData	A
ibm-idsTraceRecordSize	A
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	I
ibm-idsFSThreshold	I
ibm-idsSSInterval	I
ibm-idsSSThreshold	I
ibm-idsSensitivity	I
ibm-idsScanExclusion	I
ibm-idsIfcFloodPercentage	I
ibm-idsIfcFloodMinDiscard	I
Notes: 1. Additional values are allowed in same action. 2. SYSLOGDETAIL is equivalent to SYSLOG.	

Table 88 lists more IDS attack policies.

Table 88. IDS attack policies (RESTRICTED PROTOCOL and RAW)

Mapping conditions	
ibm-idsConditionType	"ATTACK"
ibm-idsAttackType	"RESTRICTED_IP_PROTOCOL" | "OUTBOUND_RAW"
Other conditions	
ibm-idsIPOptionRange	I
ibm-idsLocalPortRange	I
ibm-idsRemotePortRange	I
ibm-idsProtocolRange	A (1)
ibm-idsLocalHostIPAddress	I
ibm-idsRemoteHostIPAddress	I
Actions	
ibm-idsActionType	"ATTACK" (2)
ibm-idsTypeActions	A
ibm-idsNotification	A (3)

Table 88. IDS attack policies (RESTRICTED PROTOCOL and RAW) (continued)

Mapping conditions	
ibm-idsLoggingLevel	A
ibm-idsStatInterval	A
ibm-idsMaxEventMessage	A
ibm-idsTraceData	A
ibm-idsTraceRecordSize	A
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	I
ibm-idsFSThreshold	I
ibm-idsSSInterval	I
ibm-idsSSThreshold	I
ibm-idsSensitivity	I
ibm-idsScanExclusion	I
ibm-idsIfcFloodPercentage	I
ibm-idsIfcFloodMinDiscard	I

Notes:

1. If no protocol ranges are specified, no protocols are restricted. Protocols 1 (ICMP), 6 (TCP), and 17 (UDP) are treated differently for RESTRICTED_IP_PROTOCOL and OUTBOUND_RAW. They are ignored if present in a RESTRICTED_IP_PROTOCOL policy. They are obeyed if present in an OUTBOUND_RAW policy.
2. Additional values are allowed in same action.
3. SYSLOGDETAIL is equivalent to SYSLOG.

Table 89 lists more IDS attack policies.

Table 89. IDS attack policies (RESTRICTED OPTIONS)

Mapping conditions	
ibm-idsConditionType	"ATTACK"
ibm-idsAttackType	"RESTRICTED_IP_OPTIONS"
Other conditions	
ibm-idsIPOptionRange (3)	A
ibm-idsLocalPortRange	I
ibm-idsRemotePortRange	I
ibm-idsProtocolRange	I
ibm-idsLocalHostIPAddress	I
ibm-idsRemoteHostIPAddress	I
Actions	
ibm-idsActionType	"ATTACK" (1)
ibm-idsTypeActions	A
ibm-idsNotification	A (2)

Table 89. IDS attack policies (RESTRICTED OPTIONS) (continued)

Mapping conditions	
ibm-idsLoggingLevel	A
ibm-idsStatInterval	A
ibm-idsMaxEventMessage	A
ibm-idsTraceData	A
ibm-idsTraceRecordSize	A
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	I
ibm-idsFSThreshold	I
ibm-idsSSInterval	I
ibm-idsSSThreshold	I
ibm-idsSensitivity	I
ibm-idsScanExclusion	I
ibm-idsIfcFloodPercentage	I
ibm-idsIfcFloodMinDiscard	I

Notes:

1. Additional values are allowed in same action.
2. SYSLOGDETAIL is equivalent to SYSLOG.
3. If no option ranges are specified, all options are restricted. Options 0 (end of option list) and 1 (no-operation) are always allowed. They are ignored if present.

Table 90 lists more IDS attack policies.

Table 90. IDS attack policies (PERPETUAL ECHO)

Mapping conditions	
ibm-idsConditionType	"ATTACK"
ibm-idsAttackType	"PERPETUAL_ECHO" (1)
Other conditions	
ibm-idsIPOptionRange	I
ibm-idsLocalPortRange	R (1), (2)
ibm-idsRemotePortRange	R (1), (2)
ibm-idsProtocolRange	I
ibm-idsLocalHostIPAddress	I
ibm-idsRemoteHostIPAddress	I
Actions	
ibm-idsActionType	"ATTACK" (3)
ibm-idsTypeActions	A
ibm-idsNotification	A (4)
ibm-idsLoggingLevel	A

Table 90. IDS attack policies (PERPETUAL ECHO) (continued)

Mapping conditions	
ibm-idsStatInterval	A
ibm-idsMaxEventMessage	A
ibm-idsTraceData	A
ibm-idsTraceRecordSize	A
ibm-idsTRtcpTotalConnections	I
ibm-idsTRtcpPercentage	I
ibm-idsTRtcpLimitScope	I
ibm-idsTRudpQueueSize	I
ibm-idsFSInterval	I
ibm-idsFSThreshold	I
ibm-idsSSInterval	I
ibm-idsSSThreshold	I
ibm-idsSensitivity	I
ibm-idsScanExclusion	I
ibm-idsIfcFloodPercentage	I
ibm-idsIfcFloodMinDiscard	I

Notes:

1. This must be CNF with three condition levels. One condition level has ibm-idsAttackType, a second has one or more ibm-idsLocalPortRange conditions, and a third has one or more ibm-idsRemotePortRange conditions.
2. Only the first 20 ports specified is used.
3. Additional values are allowed in same action.
4. SYSLOGDETAIL is equivalent to SYSLOG.

Table 91 lists IDS traffic regulation (TR) policies.

Table 91. IDS TR policies

Mapping Conditions		
ibm-idsConditionType	"TR"	"TR"
ibm-idsProtocolRange	"6" (TCP)	"17" (UDP)
ibm-idsLocalHostIPAddress	A	A
ibm-idsLocalPortRange	A	A
Other conditions		
ibm-idsAttackType	X	X
ibm-idsIPOptionRangeLocalPortRange	X	X
ibm-idsRemotePortRange	X	X
ibm-idsRemoteHostIPAddress	X	X
Actions		

Table 91. IDS TR policies (continued)

Mapping Conditions		
ibm-idsActionType	"TR" (1)	"TR" (1)
ibm-idsTypeActions	A	A
ibm-idsNotification	A	A
ibm-idsLoggingLevel	A	A
ibm-idsStatInterval	A	A
ibm-idsMaxEventMessage	I	I
ibm-idsTraceData	A	A
ibm-idsTraceRecordSize	A	A
ibm-idsTRtcpTotalConnections	A	I
ibm-idsTRtcpPercentage	A	I
ibm-idsTRtcpLimitScope	A	I
ibm-idsTRudpQueueSize	I	A
ibm-idsFSInterval	I	I
ibm-idsFSThreshold	I	I
ibm-idsSSInterval	I	I
ibm-idsSSThreshold	I	I
ibm-idsSensitivity	I	I
ibm-idsScanExclusion	I	I
ibm-idsIfcFloodPercentage	I	I
ibm-idsIfcFloodMinDiscard	I	I
ibm-idsIfcFloodMinDiscard	I	I

Notes:

1. Additional values are allowed in same action.

Chapter 24. Simple Network Management Protocol

This topic provides information about configuring and starting the following SNMP functions:

- "SNMP agent (OSNMPD)"
- "SNMP query engine (SNMPQE)" on page 1276
- "z/OS UNIX snmp command" on page 1280
- "TRAPFWD daemon" on page 1286

SNMP agent (OSNMPD)

This topic includes information about the SNMP agent (OSNMPD).

Starting OSNMPD from MVS

If you want to start the SNMP agent by using an MVS cataloged procedure, you can use the sample cataloged procedure, OSNMPDPR. Copy member OSNMPDPR from SEZAINST to your system or recognized PROCLIB. The default name for the SNMP agent is OSNMPD. Specify the SNMP agent parameters that you need and change the data set or file names to suit your local configuration.

Sample SNMP agent cataloged procedure

Following is the sample SNMP agent cataloged procedure, OSNMPDPR. See the comments after the JCL EXEC statement for examples of how to specify configuration files, environment variables, and parameters to the SNMP agent.

```
//OSNMPD    PROC
//*
//* Sample procedure for running the z/OS UNIX SNMP agent
//*
//* z/OS Communications Server Version 2 Release 1
//* SMP/E Distribution Name: SEZAINST(EZASNDPR)
//*
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               5650-ZOS
//*               Copyright IBM Corp. 1997, 2013
//*
//* Status:       CSV2R1
//*
//OSNMPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=NOLIMIT,
//   PARM='-d 0'
//*
//*** Notes:
//*
//* - The C runtime libraries should be in the system's link list
//*   or this sample procedure will need to STEPLIB to them.
//*
//* - TCP/IP runtime libraries should also be in the system's link
//*   list.
//*
//* - OSNMPD must find the name (TCPIPJOBNAME in TCPIP.DATA) that
//*   it should be associated with. The OE function __iptcpn() is
//*   used to find this name. It is suggested that the parmlist
//*   be modified to set the environment variable
//*   RESOLVER_CONFIG to point to the correct resolver file when
//*   multiple INET Physical File Systems are started.  See the
//*   examples below.
```

```
//*
//*   If only one INET PFS will be started then /etc/resolv.conf
//*   may be used.
//*
//* - The OSNMPD agent can also be invoked from the OMVS shell as
//*   a shell command.
//*
//*** Examples for specifying configuration data sets
//*
//*  Example 1: TCPIP.DATA in partioned data set on the
//*             RESOLVER_CONFIG environment variable
//*
//*   //  PARM=(,
//*   //  'ENVAR("RESOLVER_CONFIG=//''TCPA.MYFILE(TCPDATA)''")/-d 0')
//*
//*  Example 2: TCPIP.DATA on a SYSTCPD DD statement
//*
//*   As an alternative to setting the RESOLVER_CONFIG environment
//*   variable, the SYSTCPD DD card may be specified.
//*   SYSTCPD explicitly identifies which data set is to be
//*   used to obtain the parameters defined by TCPIP.DATA
//*   when no GLOBALTCPIPDATA statement is configured.
//*   See the IP Configuration Guide for information on
//*   the TCPIP.DATA search order.
//*   The data set can be any sequential data set or a member of
//*   a partitioned data set (PDS).
//*
//*   //SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
//*
//*  Example 3: TCPIP.DATA and SNMPD.CONF in HFS files
//*
//*   //  PARM=('ENVAR("RESOLVER_CONFIG=/etc/tcpa.data"',
//*   //  '"SNMPD_CONF=/etc/snmpd.conf.tcpa")',
//*   //  '/-d 0')
//*
//*  Example 4: Specification of data sets via STDENV DD statement
//*
//*   //  PARM=('ENVAR("_CEE_ENVFILE_S=DD:STDENV")/-d 0')
//*
//*   For this method, the STDENV DD statement below must be
//*   uncommented and set to point to a data set containing
//*   settings for any environment variables.  For example, it
//*   can contain
//*
//*   RESOLVER_CONFIG=//'TCPIVP.TCPPARMS(TCPDATA)'
//*   SNMPD_CONF=//'TCPIVP.TCPPARMS(SNMPDIVP)'
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//*   The use of the STDENV DD statement works well when more than
//*   one environment variable is specified, as there is a JCL limit
//*   of 100 characters on the PARM= statement.
//*
//*** Example for specifying z/OS UNIX pathname
//*
//*   If you have configured a common INET (CINET) environment, and
//*   you want to start an SNMP agent for more than one TCP/IP stack,
//*   and you are going to enable SNMP subagents that connect to
//*   the agent using a z/OS UNIX connection, you will need to
//*   define unique z/OS UNIX pathnames for each agent.  You do
//*   this by specify the -s parameter:
//*
//*   //  PARM='-d 0 -s /var/tcpip1_dpi_socket'
//*
//*** See the SNMP chapters in the IP Configuration Guide and the
//*   IP Configuration Reference for details on the configuration
//*   files used by the SNMP agent, the search orders associated
//*   with them, and the SNMP agent start parameters.
//*
//*
//*
//*SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
//*STDENV   DD DSN=TCPIVP.TCPPARMS(SNMPENV),DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*
```

Figure 39. OSNMPD MVS started procedure

Restriction: When you use the environment variable _CEE_ENVFILE with an MVS data set, allocate the data set with the value RECFM=V. You should not use RECFM=F because RECFM=F causes the environment variable values to be padded with blanks.

Starting OSNMPD from the z/OS UNIX System Services shell

To start OSNMPD from the z/OS UNIX System Services shell, use the following syntax:

```
►►──osnmpd──┬────┬──┬────┬──┬──────────┬──┬──────────────┬──┬──────────┬──────────►
            └─-A─┘  └─-a─┘  └─-C class─┘  └─-c community─┘  └─-d level─┘

►──┬─────────────┬──┬────────┬──┬───────────────┬──┬────────┬──────────────────►◄
   └─-i interval─┘  └─-p port─┘  └─-s socketname─┘  ├─?──────┤
                                                    └─-help──┘
```

OSNMPD parameters

The SNMP agent (OSNMPD) runs in a separate address space that executes load module EZASNMPD. OSNMPD can be started without parameters or you can add any of the parameters in this topic.

When starting OSNMPD from MVS, add the parameters to the PARMS= keyword on the EXEC statement of the OSNMPD cataloged procedure. When starting OSNMPD from z/OS UNIX System Services, specify the desired parameters on the osnmpd command.

Rule: The parameters must be entered in lowercase because they are case sensitive.

Parameter	Description
-A	Forces the SNMP Agent to obtain an IPv4 address for itself when it initializes. If no IPv4 addresses are available, then the IPv4 loopback address (127.0.0.1) is used. If this parameter is not specified, the SNMP Agent uses an IP address that might be IPv6 or IPv4.

-a
Specifies that the packets sent by the SNMP Agent for responses and notifications should be sent using the physical interface address, rather than a VIPA address (if SOURCEVIPA is configured).

-C *class*
Permits you to control SNMP subagent connections to the SNMP agent via a z/OS UNIX connection. For z/OS UNIX connections, a z/OS UNIX path name is used. This path name can be specified on the agent's -s parameter. See the description of the -s parameter in this topic for more information. All of the z/OS Communications Server SNMP subagents use a z/OS UNIX connection to connect to the agent.

In order for subagents to successfully connect to the agent, either the subagents must be defined with superuser authority or the path name's read and write file access permission bits must be set for the class associated with the subagent's user ID. For more detailed information about file access permission bits, see the information about handling security for your files in z/OS UNIX System Services User's Guide.

This parameter's *class* value specifies the class or classes of the subagent user IDs, for those subagents that you want to permit to connect to the SNMP agent using a z/OS UNIX connection. This parameter causes the path name's read and write file access permission bits to be set for the specified classes.

The valid values for the *class* variable are 1 - 4 and are defined as follows:

1 Group class
Specify this value if you want only those subagents whose user IDs are associated with the same security product group as the agent to be able to connect. The resulting file access permission bit value in octal is 660.

2 Other class
Specify this value if you want only those subagents whose user IDs are not associated with the same security product group as the agent to be able to connect. The resulting file access permission bit value in octal is 606.

3 Both Group and Other class
Specify this value if you want all subagents to be able to connect to the agent. The resulting file access permission bit value in octal is 666.

4 Only User class
Specify this value if you do not want any subagents to be able to connect to the agent using a z/OS UNIX connection.

If the -C parameter is not specified, default level 1 is used. This means that the group read and write permission bits for the path name are set and that subagents whose user IDs are associated with the same security product group as the SNMP agent are able to connect.

-c *community*
This parameter can be used to dynamically configure a community name to the agent instead of defining it in the SNMP agent configuration file. A community name is a password that can accompany an SNMP request that the agent receives. Use community names with community-based security to restrict access to SNMP management data. See Configure the SNMP

agent and Configure the SNMP agent (OSNMPD) in z/OS Communications Server: IP Configuration Guide for more information about community-based security.

The value that you specify for *community* is configured as both an ASCII and an EBCDIC community name. This parameter should be specified only if you are using community-based security with the SNMP agent, and you want to dynamically define a community name that any requestor can use to retrieve SNMP management data. Specifying a community name on this parameter when starting the agent causes the community to be defined to the agent with a mask and an IP address of zeros. Therefore, any request received with this *community* value would be authenticated (for example, the request would be accepted from any IP address). If the specified community name is also defined in the SNMP agent configuration file, the definition for this community name in the configuration file is overridden by specifying the community name on the -c parameter. This parameter is case sensitive.

The default value for this parameter is the community name *public*, but this default community name is dynamically defined to the agent only if no agent configuration file is found.

-d *level*
Specifies the level of tracing to be started. The valid values for *level* are 0–255. If the -d parameter is not specified, then the default level of 0 is used, meaning no tracing is done. If the -d parameter is specified without a *level*, then a level of 31 is used, meaning all SNMP requests/responses/traps and DPI activity is traced.

There are eight levels of tracing provided. Each level selected has a corresponding number. The sum of the numbers associated with each level of tracing selected is the value which should be specified as *level*. If the agent is started, tracing options can be dynamically changed using the MVS MODIFY command. For more information about agent tracing, see the z/OS Communications Server: IP Diagnosis Guide.

The numbers for the trace levels are:

0	No tracing (default)
1	Trace SNMP requests
2	Trace SNMP responses
4	Trace SNMP traps
8	Trace DPI packets
16	Trace DPI internals (currently, no specific traces are recorded for this trace level)
32	Agent internal trace
64	Agent internal trace plus extended storage dump traces
128	Agent internal trace plus extended storage dump traces and additional information

-i *interval*
Specifies the interval (in minutes) at which dynamic configuration changes to the SNMP agent should be written out to the SNMPD.CONF

configuration file. Valid values are 0–10. The default value is 5. This parameter is only relevant when the SNMPD.CONF file is used for SNMPv3 configuration.

Guideline: Configuration updates made dynamically (by SNMP SET requests) cause the SNMPD.CONF file to be overwritten by the SNMP agent.

-p *port* Listens for SNMP packets on this port. The default is port 161. If you change the port to something other than 161, you must also configure any subagents and managers, such as osnmp, to use the new port.

-s *socketname*
Specifies the path name of the z/OS UNIX file to be used in accepting requests from subagents that communicate with the agent by way of z/OS UNIX connections. This value can be configured either by specifying it on the `-s` parameter or by specifying it as the value of the dpiPathNameForUnixStream MIB object in OSNMPD.DATA. The default is /var/dpi_socket. All of the z/OS Communications Server SNMP subagents use a z/OS UNIX connection to connect to the agent.

The SNMP agent creates this path name every time it initializes. In order for subagents to successfully connect to the agent using this path name, either the subagents must be defined with superuser authority or, the file access permission bits for this path name must be set to read and write.

- If the subagent's user ID is associated with the same security product group as the SNMP agent, the Group read and write permission bits must be set.
- If the subagent's user ID is not associated with the same security product group as the SNMP agent, the Other read and write permission bits must be set.

You can use the agent's -C parameter to ensure that the agent sets the appropriate permission bits when the agent creates the path name.

For more detailed information about file access permission bits, see information about handling security for your files in z/OS UNIX System Services User's Guide. If you need to change the path name's file access permission bits after the agent has initialized, you can use the z/OS UNIX chmod command. For more information about the chmod command, see z/OS UNIX System Services Command Reference.

? | -help
Displays the usage statement for the command. If this parameter is specified, all other parameters are ignored. If OSNMPD was started from MVS, the usage information is written to syslogd. If OSNMPD was started from z/OS UNIX System Services, the usage information is displayed to the invoker of the command.

OSNMPD environment variables

Table 92 on page 1247 provides a list of environment variables used by OSNMPD that can be tailored to a particular installation:

Table 92. OSNMPD environment variables

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules (or a link to syntax)
OSNMPD_DATA	SNMP agent	Specifies the location of the OSNMPD.DATA configuration file.	None
PW_SRC	SNMP agent	Specifies the location of the PW.SRC configuration file.	None
SNMPD_BOOTS	SNMP agent	Specifies the location of the SNMPD.BOOTS configuration file.	None
SNMPD_CONF	SNMP agent	Specifies the location of the SNMPD.CONF configuration file.	None
SNMPTRAP_DEST	SNMP agent	Specifies the location of the SNMPTRAP.DEST configuration file.	None

OSNMPD.DATA statement syntax

The OSNMPD.DATA statements specify MIB objects and their values. The format of each statement is:

```
object_name   value
```

Note:

1. There can only be one *object_name* and associated *value* per statement.
2. The *value* (if the *value* is a display or octet string) is case sensitive and is saved in mixed case.
3. Any display or octet string *value* that has imbedded white space must be enclosed in double quotation marks. For an example, see the sysDescr setting in the sample OSNMPD.DATA shipped as /usr/lpp/tcpip/samples/osnmpd.data.
4. An entry must be contained on one line.
5. Sequence numbers are not allowed on the statements.
6. Comments begin with either an asterisk (*) or a # character.

Guideline: If an error is detected in processing an entry and no appropriate default value can be assumed, the entry is discarded and an error message is written to the syslog daemon.

OSNMPD.DATA search order

The search order for accessing OSNMPD.DATA information is as follows. The first file found in the search order is used.

1. The name of a z/OS UNIX file or MVS data set specified by the OSNMPD_DATA environment variable
2. /etc/osnmpd.data z/OS UNIX file
3. The data set specified on the OSNMPD DD statement in the agent procedure
4. *jobname*.OSNMPD.DATA, where *jobname* is the name of the job used to start the SNMP agent
5. SYS1.TCPPARMS(OSNMPD)

6. *hlq*.OSNMPD.DATA, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used

If creating a data set, you can specify a sequential data set with the following attributes: RECFM=FB, LRECL=80, and BLKSZ=3120. Other data set attributes might also work, depending on your installation parameters.

OSNMPD.DATA example

A sample of OSNMPD.DATA is installed as z/OS UNIX file /usr/lpp/tcpip/samples/osnmpd.data. This sample can be modified for your installation.

```
#
# osnmpd.data sample
#
# Sample file for setting MIB variables and options for
# the SNMPv3 Agent provided by z/OS Communications Server
#
# Licensed Materials - Property of IBM
# 5650-ZOS
# Copyright IBM Corp. 1996, 2013
# Status = CSV2R1
#
  sysDescr "SNMPv3 agent version 1.0 with DPI version 2.0"
  sysContact  "Unknown"
  sysLocation "Unknown"
  sysName "z/OS V2R1 Communications Server"
# Default value of sysObjectID is equivalent to ibmTcpIpMvs
#  in the ibmAgents subtree; this is the sysObjectID representing
#  IBM z/OS Communications Server
# Changing this value is not recommended, as it is intended to allow
#  network management applications to identify this agent as the
#  z/OS Communication Server SNMP agent.  The ability to change it
#  will be disabled in a subsequent release.
# sysObjectID     "1.3.6.1.4.1.2.3.13"
  snmpEnableAuthenTraps 1
  saDefaultTimeout    6
  saMaxTimeout  700
# saAllowDuplicateIDs must be set to 1 to allow multiple DPI version 1
#  subagents
  saAllowDuplicateIDs   1
  dpiPathNameForUnixStream "/var/dpi_socket"
# Default value of sysServices indicates support for
#  internet, end-to-end, and application layers as
#  defined in RFC 1907.
  sysServices       76
```

Figure 40. OSNMPD.DATA example

PW.SRC statement syntax

The PW.SRC statements specify community names and hosts that can use each community name. The format of a statement is:

```
community_name  desired_network  snmp_mask
```

The *community_name* can be up to 32 characters in length. This value can contain both uppercase and lowercase letters; however, it is case sensitive. In any requests received by the SNMP agent, the *community_name* must match the *community_name* specified in PW.SRC exactly, including the correct case.

The *desired_network* is the IPv4 address in dotted decimal notation or IPv6 address in colon hexadecimal notation representing the range of addresses for which this *community_name* can be used. The *desired_network* must be specified; there is no default value.

If *desired_network* is an IPv6 address, then *snmp_mask* is either an IPv6 address mask in colon hexadecimal notation or an integer from 0 to 128 specifying the number of IPv6 address prefix bits used to construct an IPv6 address mask. The IP address mask is logically ANDed with the origin address of the incoming SNMP message.

If *desired_network* is an IPv4 address, then *snmp_mask* is either an IPv4 address mask (for example, 255.255.255.0) or an integer from 0 to 32 specifying the number of IPv4 address prefix bits used to construct an IPv4 address mask. The IP address mask is logically ANDed with the origin address of the incoming SNMP message.

Restriction: Scope information cannot be specified as part of the *desired_network* value.

If the value resulting from this logical ANDing equals the value of *desired_network*, the incoming message is accepted. *snmp_mask* must be specified; there is no default value.
- All parameters for each community must be on the same statement.
- Sequence numbers are not allowed on the statements.
- Comments begin with either an asterisk (*) or a # character.

Guidelines:
- If an error is detected in processing an entry and no appropriate default value can be assumed, the entry is discarded and an error message is written to the syslog daemon.

PW.SRC search order

The search order for accessing PW.SRC information is as follows. The first file found in the search order is used.
1. The name of a z/OS UNIX file or an MVS data set specified by the PW_SRC environment variable
2. /etc/pw.src z/OS UNIX file
3. The data set specified on SYSPWSRC DD statement in the agent procedure
4. *jobname*.PW.SRC, where *jobname* is the name of the job used to start the SNMP agent
5. SYS1.TCPPARMS(PWSRC)
6. *hlq*.PW.SRC, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used

Because this file can only be used with SNMPv3, you should verify that there is no SNMPD.CONF file . If an SNMPD.CONF file is found, the PW.SRC file is not used.

SNMPTRAP.DEST statement syntax

The SNMPTRAP.DEST statements list managers who are to receive the traps, and the protocol used to send traps. The format of a statement is:

```
manager UDP
```

The *manager* is the host to which the trap is to be sent. This can be a host name, or it can be the IP address of the host in IPv4 dotted decimal or IPv6 colon hexadecimal notation. If a host name is specified, the value can contain both uppercase and lowercase letters and it is not case sensitive. The protocol must be UDP. There should be one entry in the data set for each host to which you want to send traps.

- All parameters for each host must be on the same statement.
- Sequence numbers are not allowed on the statements.
- Comments begin with an asterisk (*) or a # character.

Restriction: Scope information cannot be specified as part of the *manager* value.

Guideline: If an error is detected in processing an entry and no appropriate default value can be assumed, the entry is discarded and an error message is written to the syslog daemon.

SNMPTRAP.DEST search order

The search order for accessing SNMPTRAP.DEST is as follows. The first file found in the search order is used.

1. The name of a z/OS UNIX file or an MVS data set specified by the SNMPTRAP_DEST environment variable
2. /etc/snmptrap.dest z/OS UNIX file
3. The data set specified on SNMPTRAP DD statement in the agent procedure
4. *jobname*.SNMPTRAP.DEST, where *jobname* is the name of the job used to start the SNMP agent
5. SYS1.TCPPARMS(SNMPTRAP)
6. *hlq*.SNMPTRAP.DEST, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used

Verify that there is no SNMPD.CONF file. If an SNMPD.CONF file is found, the SNMPTRAP.DEST file is not used.

SNMPD.CONF search order

The search order for accessing SNMPD.CONF information is as follows. The first file found in the search order is used.

1. The name of a z/OS UNIX file or an MVS file specified by the SNMPD_CONF environment variable.
2. /etc/snmpd.conf

If the SNMPD.CONF file is found, the PW.SRC file and the SNMPTRAP.DEST file are not used.

SNMPD.CONF statements

If you want to migrate your community-based configuration information from the PW.SRC and SNMPTRAP.DEST files to the SNMPD.CONF file, see “Migrating the PW.SRC file and SNMPTRAP.DEST file to the SNMPD.CONF file” on page 1274 for help with coding the SNMPD.CONF statements.

The SNMPD.CONF file supports the following types of entries:

USM_USER
Defines a user for the user-based security model (USM).

VACM_GROUP
Defines a security group (made up of users or communities) for the view-based access control model (VACM).

VACM_VIEW
Defines a particular set of MIB objects, called a view, for the VACM.

VACM_ACCESS
Identifies the access permitted to different security groups for the VACM.

NOTIFY
Identifies management targets to receive notifications.

NOTIFY_FILTER_PROFILE
Associates a notify filter with a particular set of target parameters.

NOTIFY_FILTER
Defines a filter profile used to filter notifications (for example, traps or informs).

TARGET_ADDRESS
Defines a management application's address and identifies parameters to be used in sending notifications or in processing requests when using community-based security.

TARGET_PARAMETERS
Defines the message processing and identifies security parameters to be used in sending notifications to a particular management target or when processing requests when using community-based security.

COMMUNITY
Defines a community for community-based security. Communities defined with this syntax are supported for compatibility purposes with the statements from the PW.SRC file, but they cannot be changed dynamically by way of SNMP SET commands. If you are defining community-based security for the first time, you should use the SNMP_COMMUNITY statement.

SNMP_COMMUNITY
Defines a community for community-based security. Communities defined with this statement can be dynamically changed by way of SNMP SET commands to the snmpCommunityTable.

The following statements must also be configured to complete the definition of a community:

- VACM_GROUP specifies the group associated with the community.
- VACM_ACCESS specifies the access allowed for the community group.
- TARGET_ADDRESS defines the address range permitted to use a particular community name and the maximum size of a response from the SNMP agent. The *transportTag* field of the SNMP_COMMUNITY statement specifies the name of the associated TARGET_ADDRESS statement.
- TARGET_PARAMETERS defines the SNMP protocol to be used with this community name.

DEFAULT_SECURITY
Identifies the default security posture configured for the SNMP agent. Additional security definitions defined by the use of the preceding eight entry definition types augment any default security configurations defined as a result of the DEFAULT_SECURITY statement.

Steps for configuring the SNMP agent for community-based security and SNMPv3 user-based security

This topic provides steps for coding the SNMPD.CONF statements required for configuring community-based security and SNMPv3 user-based security.

Steps for configuring community-based security

This topic describes the steps of configuring the SNMP agent for community-based security.

Procedure

To configure the SNMP agent for community-based security, perform the following steps:

1. For each community name, create an SNMP_COMMUNITY statement in the SNMPD.CONF file. This identifies the community names defined to the agent. For the security name field on the statement, you can use the community name, or you can define a different security name value. The value in the security name field is used to correlate the community with its associated VACM_GROUP and TARGET_PARAMETERS entries. The SNMP_COMMUNITY statement also refers to the associated TARGET_ADDRESS entry.

2. Create TARGET_ADDRESS entries to identify the address range permitted to use a particular community name. For each SNMP_COMMUNITY entry and for each range of addresses for which it is used, create a TARGET_ADDRESS entry. The TARGET_ADDRESS entries refer to related TARGET_PARAMETERS entries.

3. Create TARGET_PARAMETERS entries to identify the message processing and security models (SNMPv1 or SNMPv2c) to be used with the address on the corresponding TARGET_ADDRESS entry.

4. Define the following entries to determine which SNMP communities get access to which pieces of data and the type of access that they are allowed:

   VACM_GROUP
   Specify one entry for each community per security model (SNMPv1 or SNMPv2c). The security name field associates this group with its SNMP_COMMUNITY entry.

   VACM_VIEW
   Specify one entry for each set of MIB object identifiers that you want to protect.

   VACM_ACCESS
   Specify one entry that ties together the VACM_GROUP and VACM_VIEW entries and defines each group or view permission. You can define the group's permission to read, write, and receive notifications for the defined views.

5. To configure the managers to which traps or notifications should be sent, create the following entries:
   - Add one NOTIFY entry for each manager for each security model (SNMPv1 and SNMPv2c).

- Add one TARGET_ADDRESS statement for each manager for each security model (SNMPv1 and SNMPv2c) to define the IP addresses to which traps or notification should be sent. The TARGET_ADDRESS entry references the TARGET_PARAMETERS entry where the security model is defined.
- Add one TARGET_PARAMETERS entry to identify the security model (SNMPv1 or SNMPv2c) used in sending notifications to particular destination.
- If you want to send the trap or notification with a community name, add an SNMP_COMMUNITY statement.

Example

Figure 41 on page 1254 shows how to define the SNMPD.CONF statements for community-based security.

```
#----------------------------------------------------------------------------------
# SNMPD.CONF file for SNMP community-based security
#----------------------------------------------------------------------------------
# VACM_GROUP entries
# Format is:
#         grpName  secModel  secName     storType
#----------------------------------------------------------------------------------
# VACM_GROUP statements for SNMP community-based security requests
VACM_GROUP group1   SNMPv1    reqsnv1     -
VACM_GROUP group1   SNMPv2c   reqsnv2c    -
# VACM_GROUP statements for traps
VACM_GROUP group2   SNMPv1    trapsnv1    -
VACM_GROUP group2   SNMPv2c   trapsnv2c   -
#----------------------------------------------------------------------------------
# VACM_VIEW entries
# Format is:
#        viewName viewSubtree viewMask viewType storType
#----------------------------------------------------------------------------------
VACM_VIEW bigView  internet    -       included -
#----------------------------------------------------------------------------------
# VACM_ACCESS entries
# Format is:
#           grpName cP cM secLevel secModel readView writeView notifyView storType
#----------------------------------------------------------------------------------
VACM_ACCESS  group1  -  -  noAuthNoPriv SNMPv1   bigView  bigView   bigView    -
VACM_ACCESS  group1  -  -  noAuthNoPriv SNMPv2c  bigView  bigView   bigView    -
VACM_ACCESS  group2  -  -  noAuthNoPriv SNMPv1   bigView  bigView   bigView    -
VACM_ACCESS  group2  -  -  noAuthNoPriv SNMPv2c  bigView  bigView   bigView    -
#----------------------------------------------------------------------------------
# SNMP_COMMUNITY
# Format is:
#             commIndx   commName   secName    cI cN transTag    storType
#----------------------------------------------------------------------------------
# SNMP_COMMUNITY statements for SNMP community-based security
SNMP_COMMUNITY commreqv1  reqpwv1    reqsnv1    -  -  tagv1       -
SNMP_COMMUNITY commreqv2c reqpwv2c   reqsnv2c   -  -  tagv2c      -
# SNMP_COMMUNITY statements for traps
SNMP_COMMUNITY commtrpv1  trappwv1   trapsnv1   -  -  tagtrapv1   -
SNMP_COMMUNITY commtrpv2c trappwv2c  trapsnv2c  -  -  tagtrapv2c  -
#----------------------------------------------------------------------------------
# NOTIFY entries
# Format is:
#      notifyName tag         type  storType
#----------------------------------------------------------------------------------
NOTIFY notify1    tagtrapv1   trap  -
NOTIFY notify2    tagtrapv2c  trap  -
#----------------------------------------------------------------------------------
# TARGET_ADDRESS
# Format is:
#           taName  Dom tAddress   tagList    taParms tI rC sT tMask          tMMS
#----------------------------------------------------------------------------------
# TARGET_ADDRESS statements for SNMP community-based security
```

```
TARGET_ADDRESS targad1 UDP ipaddress  tagv1  targp1  -  -  -  255.255.255.255..0 65535
TARGET_ADDRESS targad2 UDP ipaddress  tagv2c targp2  -  -  -  255.255.255.255..0 65535
# TARGET_ADDRESS statements for traps
TARGET_ADDRESS targad3 UDP ipaddress  tagtrapv1  targp3  -  -  -  -              -
TARGET_ADDRESS targad4 UDP ipaddress  tagtrapv2c targp4  -  -  -  -              -
#----------------------------------------------------------------------------------
# TARGET_PARAMETERS
# Format is:
#               pName Model    secModel secName    secLevel     storType
#----------------------------------------------------------------------------------
# TARGET_PARAMETERS statements for SNMP community-based security
TARGET_PARAMETERS targp1 SNMPv1  SNMPv1   reqsnv1    noAuthNoPriv -
TARGET_PARAMETERS targp2 SNMPv2c SNMPv2c  reqsnv2c   noAuthNoPriv -
# TARGET_PARAMETERS statements for traps
TARGET_PARAMETERS targp3 SNMPv1  SNMPv1   trapsnv1   noAuthNoPriv -
TARGET_PARAMETERS targp4 SNMPv2c SNMPv2c  trapsnv2c  noAuthNoPriv -
```

Figure 41. Example of SNMPD.CONF statements for community-based security

Steps for configuring SNMPv3 user-based security

This topic describes the steps of configuring the SNMP agent for SNMPv3 user-based security.

Procedure

To configure the SNMP agent for SNMPv3, perform the following steps to determine which SNMPD.CONF statements you need to specify:

1. Determine which SNMP users (typically managers) communicate SNMP requests to the agent. Define the USM_USER statement to define the name, the protocol, and key to authenticate messages for the user, and the protocol and key to encrypt messages for the user. The command can be used to generate the keys.

2. Determine which SNMP users get access to which pieces of data, and the type of access they are allowed. Define the following VACM_* statements:

   VACM_GROUP
   To identify members of a group with the same access privileges. The security model for SNMPv3 is USM.

   VACM_VIEW
   To identify the MIB object identifiers to which access is permitted or denied.

   VACM_ACCESS
   To tie together the VACM_GROUP and VACM_VIEW statements by defining the group's permission to read, write, and receive notifications for the defined views.

3. To send notifications, define the following configuration statements:
   - Add one NOTIFY statement for each type of notification, such as TRAP or INFORM notifications.
   - Add one TARGET_ADDRESS entry for each manager that receives a notification.
   - Optionally, configure a TARGET_PARAMETERS entry to define the security parameters, such as encryption and authentication, used when sending notifications

Coding the SNMPD.CONF entries

This topic describes how to code the SNMPD.CONF entries.

Defining the USM_USER entry

```
►►──USM_USER──userName──engineID──authProto──authKey──────────────────►

►─privProto──privKey──keyType──storageType─────────────────────────────►◄
```

Defining the VACM_GROUP entry

```
►►──VACM_GROUP──groupName──securityModel──securityName──storageType────►◄
```

Defining the VACM_VIEW entry

```
►►──VACM_VIEW──viewName──viewSubtree──viewMask──viewType──storageType──►◄
```

Defining the VACM_ACCESS entry

```
►►──VACM_ACCESS──groupName──contextPrefix──contextMatch──securityLevel──►

►─securityModel──readView──writeView──notifyView──storageType──────────►◄
```

Defining the NOTIFY entry

```
►►──NOTIFY──notifyName──tag──type──storageType─────────────────────────►◄
```

Defining the NOTIFY_FILTER_PROFILE entry

```
►►──NOTIFY_FILTER_PROFILE──targetParamsName──profileName──storageType──►◄
```

Defining the NOTIFY_FILTER entry

```
►►──NOTIFY_FILTER──profileName──filterSubtree──filterMask──filterType──storageType──►◄
```

Defining the TARGET_ADDRESS entry

```
►►──TARGET_ADDRESS──targetAddrName──tDomain──tAddress──tagList─────────►

►─targetParams──timeout──retryCount──storageType──tMask──tMMS──────────►◄
```

Defining the TARGET_PARAMETERS entry

```
►►──TARGET_PARAMETERS──paramsName──mpModel──securityModel──────────────►
```

```
►─securityName──securityLevel──storageType──────────────────────────────────►◄
```

Defining the COMMUNITY entry

```
►►──COMMUNITY──communityName──securityName──securityLevel──netAddr──────────►

►─netMask──storageType──────────────────────────────────────────────────────►◄
```

Defining the SNMP_COMMUNITY entry

```
►►──SNMP_COMMUNITY──communityIndex──communityName──securityName─────────────►

►─contextEngineID──contextName──transportTag──storageType───────────────────►◄
```

Defining the DEFAULT_SECURITY entry

```
►►──DEFAULT_SECURITY──securityPosture──password──privacy────────────────────►◄
```

Parameters

USM_USER entry

`USM_USER userName engineID authProto authKey privProto privKey keyType storageType`

Field definitions

userName
: Indicates the name of the user for the User-Based Security Model (USM). The userName must be unique to the SNMP agent. The userName is used as the security name for the User-based Security Model; the contents of this field are used as the securityName value for other entries (such as the VACM_GROUP entry) when the securityModel is USM. The userName field is a 1–32 character string. There is no default value.

engineID
: Indicates the engineID of the authoritative side of the message. The engineID for the z/OS Communications Server SNMP agent is determined at agent initialization; it is either read in from the SNMPD.BOOTS file or it is generated automatically and stored in the SNMPD.BOOTS file. It can be retrieved dynamically by issuing a get request for object snmpEngineID.0. Use the following information to determine which engineID should be specified for a user:

  - For get, getbulk, set, response, and trap messages, the authoritative side is the SNMP agent. Therefore, either specify its engineID or use a dash (-) for the default value.
  - For inform messages, the authoritative side is the notification receiver. Therefore, you must configure the engineID of the notification receiver.

  Valid values are a string of 2–64 hexadecimal digits or a dash (-) to indicate the default value (the local SNMP agent's engineID).

authProto
: Indicates the authentication protocol to be used on authenticated messages on behalf of this user. The following values are valid:

- HMAC-MD5
- HMAC-SHA
- none. Indicates that no authentication is to be done.
- dash (-). Indicates the default value, which is HMAC-MD5.

authKey
: Indicates the authentication key to be used in authenticating messages on behalf of this user. This field is ignored when authProto is specified as none. The keyType field indicates whether the key is localized or nonlocalized. Valid values are 32 hexadecimal digits when authProto is HMAC-MD5 and 40 hexadecimal digits when authProto is HMAC-SHA. A (-) dash indicates the default (no key, indicating no authentication). For information about generating authentication and privacy keys using the pwtokey command, see z/OS Communications Server: IP System Administrator's Commands.

privProto
: Indicates the privacy protocol to be used on encrypted messages on behalf of this user. Privacy can be requested only if authentication is also requested. If authentication is not requested, this field is ignored. The following values are valid:

  DES
  : Indicates CBC 56-bit DES.

  AESCFB128
  : Indicates AES 128-bit CFB mode.

  none
  : Indicates no privacy.

  dash (-)
  : Indicates the default value, which is no privacy.

  Requirement: For the AES privacy protocol, z/OS Integrated Cryptographic Service Facility (ICSF) must be active. For detailed information about configuring ICSF, see z/OS Cryptographic Services ICSF Administrator's Guide.

privKey
: Is used in authenticating messages to and from this user. This field is ignored when privProto is specified as none. The keyType field indicates whether the key is localized or nonlocalized. Privacy can be requested only if authentication is also requested. If authentication is not requested, this field is ignored. The privacy key and the authentication key are assumed to have been generated using the same authentication protocol (HMAC-MD5 or HMAC-SHA). Valid values are 32 hexadecimal digits if the key is localized or if the key is nonlocalized and the authProto is HMAC-MD5; 40 hexadecimal digits if the key is nonlocalized and the authProto is HMAC-SHA; or a dash (-) to indicate the default of no key, indicating no encryption. For information about generating privacy keys using the pwtokey command, see z/OS Communications Server: IP System Administrator's Commands.

keyType
: Indicates whether the keys defined by *authKey* and *privKey* are localized or nonlocalized. Localized indicates that they have been generated with the appropriate engineID, making the key usable only at one snmpEngine. If you use this user to send inform messages and localize the key, you must

use the engineID of the notification receiver in the localized key. The notification receiver must define the user keys as nonlocalized. A nonlocalized key indicates that the key can be used at different snmpEngines. The *authKey* and *privKey*, if both are specified, must both be localized or both be nonlocalized. This field is ignored if no authentication or privacy is requested. Valid values are L to indicate keys are localized, N to indicate keys are nonlocalized, or a dash (-) to indicate the default value of localized.

storageType
Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

nonVolatile
Indicates the entry definition persists across reboots of the SNMP agent, but it can, however, be changed or even deleted by dynamic configuration requests.

permanent
Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

readonly
Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests. For the USM_USER entry, readOnly is not allowed unless the authentication protocol is none because protocols require that user keys be changeable.

dash (-)
Indicates the default value of nonVolatile.

VACM_GROUP entry

```
VACM_GROUP groupName securityModel securityName storageType
```

Field definitions

groupName
The group name for the View-Based Access Control Model (VACM). The *groupName* must be specified; there is no default value. It must be a 1–32 character string.

securityModel
Indicates the SNMP security model for this entry. When an SNMP message comes in, the *securityModel* and the *securityName* are used to determine the group to which the user (or community) represented by the *securityName* belongs. Valid values are SNMPv1 to indicate community-based security using SNMPv1 message processing; SNMPv2c to indicate community-based security using SNMPv2c message processing; USM to indicate the User-Based Security Model; or a dash (-) to indicate the default value of USM.

securityName
Indicates a member of this group. Valid values are 1–32 characters and indicate one of the following members:

- A USM userName when *securityModel* is USM
- A *securityName* from an SNMP_COMMUNITY statement whose community is associated with this group

- A *securityName* from a COMMUNITY statement whose community is associated with this group. In this case the *securityName* is the community name.

There is no default value.

storageType
Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

nonVolatile
Indicates the entry definition persists across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.

permanent
Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

readonly
Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

dash (-)
Indicates the default value of nonVolatile.

VACM_VIEW entry

```
VACM_VIEW viewName viewSubtree viewMask viewType storageType
```

Field definitions

viewName
Indicates the textual name of the view for the View-Based Access Control Model. View names do not need to be unique. Multiple entries with the same name together define one view. However, the *viewName*, together with the subtree object ID, must be unique to an SNMP engine. Valid values are 1–32 characters in length. There is no default value.

viewSubtree
Indicates the MIB object prefix of the MIB objects in the view. Valid values are an object ID of up to 128 sub-OIDs, a textual object name (or object prefix), or a combination of textual object name followed by numeric sub-OIDs. The name must be found within the compiled MIB or in the logical extension to the MIB, the /etc/mibs.data file. There is no default value.

Guideline: For views that govern notify operations (traps or informs), the viewSubtree and viewMask are used to verify access to all the MIB objects in the notification, and access to the notification OID (for example, the value in the snmpTrapOID MIB object). All notifications that the SNMP agent sends include the following MIB objects:

- sysUpTime
- snmpTrapOID
- snmpTrapEnterprise

Therefore, the most granular viewSubtree that can be specified for notifications is internet (an OID of 1.3.6.1) to permit the members of a group access to these standard notification MIB objects.

viewMask
: Indicates a mask that specifies which of the sub-OIDs in the subtree are relevant. See RFC 3415 for further information about the *viewMask*. Valid values are a hex string of up to 16 bytes (up to 128 bits), where each hexadecimal digit represents four bits. Each bit indicates whether or not the corresponding sub-OID in the subtree is relevant, or a dash (-) to indicate the default value (a mask of all ones meaning all sub-OIDs are relevant).

viewType
: Indicates the type of the view definition. Valid values are included to indicate the MIB objects identified by this view definition are within the view, excluded to indicate the MIB objects identified by this view definition are excluded from the view, or a dash (-) to indicate the default value of included.

storageType
: Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

  nonVolatile
  : Indicates the entry definition persists across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.

  permanent
  : Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

  readonly
  : Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

  dash (-)
  : Indicates the default value of *nonVolatile*

VACM_ACCESS entry

```
VACM_ACCESS groupName contextPrefix contextMatch securityLevel
securityModel readView writeView notifyView storageType
```

Field definitions

groupName
: The group name for the View-Based Access Control Model (VACM) for which access is being defined. The *groupName* must be specified; there is no default value. It must be a 1–32 character string.

contextPrefix
: Indicates a character string to be compared with the incoming contextName, if the value specified for the contextMatch field is prefix. Note, however, that the SNMP agent in z/OS Communications Server supports MIB objects in only the local (null) context. Valid values are 1–32 characters, or a dash (-) to indicate the default value of the null context ("").

contextMatch
: Indicates whether the incoming contextName must be compared with (and match exactly) the entire contextName or whether only the first part of the contextName (up to the length of the indicated value of the contextPrefix) must match. Valid values are exact to indicate that the entire contextName must match, prefix to indicate that only the prefix of the contextName must match, or a dash (-) to indicate the default value of exact.

securityLevel
: Indicates the securityLevel for this entry and is used in determining which access table entry to use. Valid values are noAuthNoPriv or none to indicate no authentication or privacy protocols are applied; AuthNoPriv or auth to indicate authentication protocols are applied but no privacy protocol is applied; AuthPriv or priv to indicate both authentication and privacy protocols are applied; or a dash (-) to indicate the default value of noAuthNoPriv.

securityModel
: Indicates the SNMP security model for this entry and is used in determining which access table entry to use. Valid values are SNMPv1 to indicate community-based security using SNMPv1 message processing, SNMPv2c to indicate community-based security using SNMPv2c message processing, USM to indicate the User-Based Security Model, or a dash (-) to indicate the default value of USM.

readView
: Indicates the name of the view to be applied when read operations (get, getnext, getbulk) are performed under control of this entry in the access table. Valid values are 1–32 characters identifying a view defined by a VACM_VIEW definition or a dash (-) to indicate the default value of no view (no readView defined for members of this group). If no view is defined, read operations fail with access authorization errors for the members of the group.

writeView
: Indicates the name of the view to be applied when write operations (set) are performed under control of this entry in the access table. Valid values are 1–32 characters identifying a view defined by a VACM_VIEW definition or a dash (-) to indicate the default value of no view (no writeView defined for members of this group). If no view is defined, write operations fail with access authorization errors for the members of the group.

notifyView
: Indicates the name of the view to be applied when notify operations (traps or informs) are performed under control of this entry in the access table. Valid values are 1–32 characters identifying a view defined by a VACM_VIEW definition or a dash (-) to indicate the default value of no view (no notifyView defined for members of this group). If no view is defined, notify operations fail with access authorization errors for the members of the group.

storageType
: Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

nonVolatile
: Indicates the entry definition persists across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.

permanent
: Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

readonly
: Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

dash (-)
: Indicates the default value of nonVolatile.

NOTIFY entry

```
NOTIFY notifyName tag type storageType
```

Field definitions

notifyName
: A locally unique identifier for this notify definition. Valid values are 1–32 characters in length. There is no default value.

tag
: Indicates a tag value to be compared with the values in the tagLists defined in the snmpTargetAddrTable (either on TARGET_ADDRESS entries or by way of dynamic configuration). For each match of this tag with a value in the tagLists defined in the snmpTargetAddrTable, a notification can be sent. See RFC 2573 for a definition of SnmpTagValue. Valid values are 1 - 255 characters. No delimiters are allowed. A dash (-) indicates the default, which is no tag value.

type
: Indicates which type of notification should be generated. Valid values are:
- A trap; an unconfirmed notification; notification sent with trap PDUs
- An inform; a confirmed notification; notification sent with inform PDUs
- dash (-) Indicates the default value of trap

storageType
: Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

nonVolatile
: Indicates the entry definition persists across reboots of the SNMP agent, but it can, however, be changed or even deleted by dynamic configuration requests.

permanent
: Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

readonly
: Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

dash (-)
Indicates the default value of nonVolatile.

NOTIFY_FILTER_PROFILE entry

`NOTIFY_FILTER_PROFILE targetParamsName profileName storageType`

Field definitions

targetParamsName
Indicates the name of the target parameter definition (paramsName in the TARGET_PARAMETERS entry) for which the specified notify filter profile is used. Valid values are 1 - 32 characters in length. There is no default value.

profileName
Indicates the name of the notify filter profile (profileName on the NOTIFY_FILTER entry) used. Valid values are 1–32 characters in length. There is no default value.

storageType
Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

nonVolatile
Indicates the entry definition persists across reboots of the SNMP agent, but it can, however, be changed or even deleted by dynamic configuration requests.

permanent
Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

readonly
Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

dash (-)
Indicates the default value of nonVolatile.

NOTIFY_FILTER entry

`NOTIFY_FILTER profileName filterSubtree filterMask filtertype storageType`

Field definitions

profileName
Indicates the name of the filter profile defined by this entry. Valid values are 1–32 characters. There is no default value.

filterSubtree
Identifies the MIB subtree that, when combined with the corresponding filterMask, defines a family of subtrees which are included in or excluded from the filter profile. Valid values are an object ID of up to 128 sub-OIDs or a textual object name (or object prefix). There is no default value.

filterMask
Indicates the bit mask that, in combination with the corresponding filterSubtree, defines a family of subtrees that are included in or excluded from the filter profile. See RFC 2573 for a definition of the viewMask. Valid

values are a hex string of up to 16 octets (up to 128 bits). Each bit indicates whether or not the corresponding subtree sub-OID is relevant, or a dash (-) to indicate the default value (a mask of all ones meaning that all sub-OIDs are relevant). inform indicates a confirmed notification (for example, notification sent with inform PDUs).

filterType
: Indicates whether the family of filter subtrees defined by this entry are included in or excluded from a filter. Valid values are included to indicate the MIB objects identified by this definition are within the filter, excluded to indicate the MIB objects identified by this definition are excluded from the filter, or a dash (-) to indicate the default value of included.

storageType
: Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

  nonVolatile
  : Indicates the entry definition persists across reboots of the SNMP agent, but it can, however, be changed or even deleted by dynamic configuration requests.

  permanent
  : Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

  readonly
  : Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

  dash (-)
  : Indicates the default value of nonVolatile.

TARGET_ADDRESS entry

TARGET_ADDRESS targetAddrName tDomain *tAddress tagList* **targetParams timeout retryCount storageType** *tMask* **tMMS**

Field definitions

targetAddrName
: Indicates a locally unique identifier for this target address definition. Valid values are 1–32 characters in length. There is no default value.

tDomain
: Indicates the transport type of the address indicated by *tAddress*. Valid values are UDP, UDP6, or a dash (-) for the default value of UDP. If *tAddress* is an IPv6 colon hexadecimal address, the value UDP6 must be used; otherwise, if *tAddress* is an IPv4 dotted decimal address, then UDP or a dash (-) must be used.

tAddress
: For notifications, *tAddress* indicates the transport address and port to which notifications are sent. For community-based security, *tAddress* indicates the value that is used to determine whether the IP address and port of an SNMP request should be permitted to the community name.

  - For IPv4 *tAddress* values, specify an IPv4 dotted decimal address that is optionally followed by two periods and a port number (for example, 10.10.1.1..1162). You can use a colon instead of two periods to separate the IPv4 address value from the port number (for example, 10.10.1.1:1162).

- For IPv6 *tAddress* values, specify an IPv6 address in colon hexadecimal notation that is optionally followed by two periods and a port number (for example, 2001:0db8::1..1162).

For community-based security, when an SNMP request is received, the IP address mask portion of the *tMask* value is logically ANDed with the origin address of the request. The resulting value must equal the value of *tAddress* for the SNMP request to be permitted to the community name. The *tagListvalue* is used to find the community name.

tagList For notifications, *tagList* indicates a list of tag values that are used to select target addresses for a notification operation. For community-based security, the *tagList* value is used to find the associated SNMP_COMMUNITY statement containing the community name used for verification of the IP address of an SNMP request. The z/OS Communications Server implementation supports, by way of the configuration file, only one tag in a tag list. RFC 2573 contains the complete definition of SnmpTagList and SnmpTagValue. The z/OS Communications Server implementation accepts as valid values a string of 1–255 characters. No delimiters are allowed. A dash (-) indicates the default value, an empty list.

targetParams
For notifications, *targetParams* indicates a TARGET_PARAMETERS *paramsName* value that indicates which security and message processing is to be used when sending notifications to this target. For community-based security, *targetParams* indicates the TARGET_PARAMETERS *paramsName* value that contains the security module to be used to determine whether an SNMP request should be permitted to a community name. Valid values are 1–32 characters in length. There is no default value.

timeout
Indicates the expected maximum round-trip time for communicating with this target address (in 1/100ths of a second). Valid values are 0 - 2147483647, or a dash (-) to indicate the default value of 1500. Timeout is used only for inform type notifications; it is not used for traps.

retryCount
Indicates the number of retries to be attempted when a response is not received for a generated message. Valid values are 0 - 255, or a dash (-) to indicate the default value of 3. RetryCount is used only for inform type notifications; it is not used for traps.

storageType
Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

nonVolatile
Indicates the entry definition persists across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.

permanent
Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

readonly
: Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

dash (-)
: Indicates the default value of nonVolatile.

tMask Indicates the IP address mask and port mask associated with this Target Address entry. If *tAddress* is an IPv4 dotted decimal address, then *tMask* must be either an IPv4 address mask in dotted decimal notation or an IPv4 prefix value (0-32), optionally followed by two periods and a port mask (for example, 255.255.0.0..65535). A colon can be used instead of two periods to separate the IPv4 address mask or prefix value from the port mask if a port mask is specified (for example, 255.255.0.0:65535). If *tAddress* is an IPv6 colon hexadecimal address, then *tMask* must be either an IPv6 address mask in colon hexadecimal notation or an IPv6 prefix value (0-128), optionally followed by two periods and a port mask (for example, 48..65535). A dash (-) specified for *tMask* indicates the default value. If tAddress is an IPv4 dotted decimal address, the default *tMask* value is 255.255.255.255..65535. If *tAddress* is an IPv6 colon hexadecimal address, the default *tMask* value is 128..65535.

Guideline: The default port mask is 65535, regardless if the port is specified in the *tAddress* field.

tMMS Indicates the maximum message size value associated with this target address entry. Valid values are in the range 0 - 2147483647. A dash (-) indicates the default, which is 484. When the TARGET_ADDRESS statement is used as part of the definition of an SNMP community name, this parameter controls the size of the response from the SNMP agent.

TARGET_PARAMETERS entry

TARGET_PARAMETERS *paramsName mpModel securityModel securityName securityLevel* **storageType**

Field definitions

paramsName
: A locally unique identifier for this target parameters definition. Valid values are 1–32 characters in length. There is no default value.

mpModel
: For notifications, *mpModel* is the message processing model to be used in sending notifications to targets with this parameter definition. For community-based security, *mpModel* designates the SNMP protocol of the SNMP requests received from the IP address defined by the associated TARGET_ADDRESS statement. Valid values are SNMPv1, SNMPv2c, and SNMPv3. There is no default value.

securityModel
: For notifications, *securityModel* indicates the security model to be used in sending notifications to targets with this parameter definition. For community-based security, *securityModel* designates the SNMP protocol of the SNMP request to be used, along with the IP address, to determine whether the IP address should be permitted to the community name. Valid values are SNMPv1, SNMPv2c, or USM to indicate User-Based Security Model. There is no default value.

securityName
: For notifications, *securityName* identifies the principal (user or community) on whose behalf SNMP messages are generated using this parameter definition. For community-based security, this is the *securityName* value from the associated SNMP_COMMUNITY statement. For user-based security, this is a *userName* value from a USM_USER statement. Valid values are 1 - 32 characters in length. There is no default value.

securityLevel
: Indicates the security level to be used in sending notifications to targets with this parameter definition. Valid values are noAuthNoPriv or none to indicate no authentication or privacy protocols are applied, AuthNoPriv or auth to indicate that authentication protocols are applied but no privacy protocol is applied, AuthPriv or priv to indicate that both authentication and privacy protocols are applied (If the additional encryption product is not applied, this level can be configured, but not actually used), or a dash (-) to indicate the default value of noAuthNoPriv. For community-based security, specify the default value of a dash(-).

storageType
: Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

  nonVolatile
  : Indicates the entry definition persists across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.

  permanent
  : Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

  readonly
  : Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

  dash (-)
  : Indicates the default value of nonVolatile.

COMMUNITY entry

`COMMUNITY` *`communityName securityName securityLevel netAddr netMask storageType`*

Field definitions

communityName
: The community name for community-based security (SNMPv1 or SNMPv2c). The *netAddr* must be specified; there is no default value. It must be a 1 - 32 character string.

securityName
: The *securityName* defined for this communityName. The *securityName* is the more generic term for the principal (user or community) for which other entries, such as VACM_GROUP and TARGET_PARAMETERS, are defined. The community name must match the *securityName* exactly. The *securityName* is 1 - 32 characters. A dash (-) indicates the default value; a *securityName* equal to the specified communityName.

securityLevel
: Indicates the security level to be applied when processing incoming or outgoing messages with this community name. Valid values are noAuthNoPriv or none to indicate no authentication or privacy protocols are applied, or dash (-) to indicate the default value of noAuthNoPriv. Encryption is not supported on SNMPv1 and SNMPv2c messages.

netAddr
: The host IP address or network address, in IPv4 dotted decimal or IPv6 colon hexadecimal notation, representing the range of addresses for which this community name can be used. When an SNMP request is received, the *netMask* value is logically ANDed with the origin address of the request. The resulting value must equal the value of *netAddr* for the SNMP request to be permitted to the community name. The netAddr must be specified; there is no default value.

netMask
: The IP address mask or IP address prefix defined for this communityName. If *netAddr* is an IPv6 colon hexadecimal address, then *netMask* is either an IPv6 colon hexadecimal address mask (for example, FFFF:FFFF::) or an integer from 0 to 128 specifying the number of IPv6 address prefix bits used to construct an IPv6 address mask. If *netAddr* is an IPv4 dotted decimal address, then *netMask* is either an IPv4 address mask (for example, 255.255.255.0) or an integer from 0 to 32 specifying the number of IPv4 address prefix bits used to construct an IPv4 address mask. The mask is logically ANDed with the origin address of the incoming SNMP message. If the resulting value equals the value of *netAddr*, the incoming message is accepted. *netMask* must be specified; there is no default value.

storageType
: Indicates the type of storage in which this definition is to be maintained. Storage types are defined in RFC 1903. Note that the value of volatile is not supported in the SNMPD.CONF file. Valid values are:

  nonVolatile
  : Indicates the entry definition persists across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.

  permanent
  : Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

  readonly
  : Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

  dash (-)
  : Indicates the default value of nonVolatile.

SNMP_COMMUNITY entry

```
SNMP_COMMUNITY communityIndex communityName securityName contextEngineID
contextName transportTag storageType
```

Field definitions

communityIndex
: Indicates a locally unique identifier for this SNMP_Community definition. Valid values are 1 - 32 characters in length. There is no default value.

communityName
: The community name for community-based security (SNMPv1 or SNMPv2c) . Valid values are 1 - 32 characters in length. There is no default value. The agent assumes that community names are encoded in ASCII. If an EBCDIC-encoded community name is needed, it must be specified as a UTF-8 name. See the Usage Note topic for an explanation about how to configure a UTF-8 name.

securityName
: The securityName defined for this *communityName*. The *securityName* is the more generic term for the principal (user or community) for which other entries, such as VACM_GROUP and TARGET_PARAMETERS, are defined. Valid values are 1 - 32 characters in length. There is no default value.

contextEngineID
: Indicates the location of the context in which information is accessed. A dash (-) indicates the default value of the local SNMP agent's engine ID. Only the default value is supported.

contextName
: The corresponding context value. Valid values are 1-32 characters in length. Only a dash (-) indicating the default, which is an empty string, is supported.

transportTag
: Indicates a tag value to be compared with the values in the tagLists defined in the snmpTargetAddrTable (either on TARGET_ADDRESS entries or by way of dynamic configuration). Those target addresses (whose tag value match this tag) identify the transport endpoints from which a request containing this community are accepted. Valid values are 1 - 255 characters. No delimiters are allowed. A dash (-) indicates the default, which is no tag value.

storageType
: Indicates the type of storage in which this definition is to be maintained. Valid values are:

  nonVolatile
  : Indicates the entry definition persists across reboots of the SNMP agent; it can, however, be changed or even deleted by dynamic configuration requests.

  permanent
  : Indicates the entry definition persists across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

  readonly
  : Indicates the entry definition persists across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests.

  dash (-)
  : Indicates the default value of nonVolatile.

DEFAULT_SECURITY entry

DEFAULT_SECURITY *securityPosture password privacy*

Field definitions

securityPosture

Indicates the default security posture to be configured for the SNMP agent, as defined by Appendix A of RFC 2575. Valid values are minimum-secure to indicate the SNMP agent is configured with the least secure default configurations; semi-secure to indicate the SNMP agent is configured with moderately secure default configurations; and no-access to indicate the SNMP agent is configured with no default configurations. The default value is no-access.

Following are the default security definitions based on the selected security posture:

no-access

No initial configurations are done.

semi-secure

If privacy is not requested, a default user is configured as if the following USM_USER entry had been specified:

```
USM_USER initial- HMAC-MD5 ### none - N permanent
```

where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

If privacy is requested, a default user is configured as if the following USM_USER entry had been specified:

```
USM_USER initial - HMAC-MD5 ### DES ### N permanent
```

where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

A default group is configured as if the following VACM_GROUP entry had been specified:

```
VACM_GROUP initial USM initial readOnly
```

Three default access entries are configured as if the following VACM_ACCESS entries had been specified:

```
VACM_ACCESS initial - exact none USM restricted -        restricted readOnly
VACM_ACCESS initial - exact auth USM internet internet  internet   readOnly
VACM_ACCESS initial - exact priv USM internet internet  internet   readOnly
```

Two default MIB views are configured as if the following VACM_VIEW entries had been specified:

```
VACM_VIEW internet   internet     - included readOnly
VACM_VIEW restricted system       - included readOnly
VACM_VIEW restricted snmp         - included readOnly
VACM_VIEW restricted snmpEngine   - included readOnly
VACM_VIEW restricted snmpMPDStats - included readOnly
VACM_VIEW restricted usmStats     - included readOnly
```

minimum-secure

If privacy is not requested, a default user is configured as if the following USM_USER entry had been specified:

```
USM_USER initial - HMAC-MD5 ### none - N permanent
```

where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

If privacy is requested, a default user is configured as if the following USM_USER entry had been specified:

```
USM_USER initial - HMAC-MD5 ### DES ### N permanent
```

where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

A default group is configured as if the following VACM_GROUP entry had been specified:

```
VACM_GROUP initial USM initial readOnly
```

Three default access entries are configured as if the following VACM_ACCESS entries had been specified:

```
VACM_ACCESS initial - exact none USM restricted -      restricted readOnly
VACM_ACCESS initial - exact auth USM internet internet internet   readOnly
VACM_ACCESS initial - exact priv USM internet internet internet   readOnly
```

Two default MIB views are configured as if the following VACM_VIEW entries had been specified:

```
VACM_VIEW internet   internet     - included readOnly
VACM_VIEW restricted internet     - included readOnly
```

password
Indicates the password to be used to generate authentication and privacy keys for user initial. If no-access is specified as the *securityPosture*, this keyword is ignored. Valid values are 8 - 255 characters string, or a dash (-) to indicate the default value (no password). The default is only accepted if *securityPosture* is no-access.

privacy Indicates whether or not encryption is to be supported for messages on behalf of user 'initial'. Valid values are Yes to indicate that privacy is supported for user initial, No to indicate that privacy is not supported for user initial, or a dash (-) to indicate the default value of no. If no-access is selected as the security posture, this value is ignored.

Usage notes

- The SNMP Agent supports the ASCII character set and UTF-8 encoding for the values specified for the name fields on the configuration statements. Use UTF-8 values to define EBCDIC name field values to the agent. To specify a UTF-8 value for a name field, specify the value in the following format:
  - The first character must be a cent sign (¢)
  - The remaining characters are the octet string representing the name value

  For example, to define the EBCDIC value, TEST, in a name field, specify the name value as follows:

  ```
  ¢e3c5e2e3
  ```
- If an error is detected in processing an entry and no appropriate default value can be assumed, the entry is discarded and an error message is written to the syslog daemon.
- An entry must be contained on one line.
- Keywords in the SNMPD.CONF file are accepted in any case (that is, they are not case sensitive.)
- Values in the SNMPD.CONF file are case sensitive. For example, a userName of user1 is not equivalent to a userName of USER1.

- All of these entry definitions require that all fields be specified, either with a specific value or with a dash (-). A dash indicates that the appropriate default value should be applied.
- If an error is detected in processing an entry and no appropriate default value can be assumed, the entry is discarded and an error message is generated.
- Statements in the SNMPD.CONF file are not order dependent. However, if more than one DEFAULT_SECURITY statement is found, the last one in the file is the one that is used.
- If there are no valid entries in the SNMPD.CONF file (but the file exists) and no -c parameter specified at agent invocation, no requests are accepted.
- Comments can be entered in the SNMPD.CONF file. They must must begin with an asterisk (*) or a # character in column 1.
- The SNMP agent uses a ¢ character to precede a hex string that represents a value for an SnmpAdminString syntax object for which the value cannot be printed. Use of the ¢ is reserved for this purpose. Do not change the contents of the entries in the configuration file that have a ¢ character preceding them.
- You cannot specify scope information about any values in the SNMPD.CONF file that represent IP addresses or host names.

SNMPD.CONF sample

The following code shows the SNMPD.CONF sample:

```
# snmpd.conf sample
#
# Sample file showing format of configuration file for the SNMP agent
#
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# 5650-ZOS
# Copyright IBM Corp. 1997, 2013
# Status = CSV2R1
# SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZASNDCO
#
#-----------------------------------------------------------------------------------------------------------------------
# Notes
# - All values for an entry must be on the same line.
# - Dynamic changes to the SNMP agent configuration made by SNMP SET
#   commands may cause entries to be written to this file that are
#   wider than the original entries.  If this file is maintained in
#   an MVS data set, the record length should allow for longer entries.
# - All localized keys need to be regenerated using the pwtokey command
#   in order for these sample entries to actually be used.
# - In this sample:
#   - Keys are generated for use with engineID 00000002000000000943714F
#   - Authentication keys were generated with password of
#     username+"password", such as "u1password"
#   - Privacy keys were generated with password of
#     username+"privpass", such as "u1privpass"
#-----------------------------------------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------------------------------------
# USM_USER entries
# Format is:
#  userName engineID authProto authKey privProto privKey keyType storageType
#
# Note: Users u3 and u4 use non-localized keys. Not recommended, but allowed.
# Note: Users u5 and u6 use the same password for generating the authkey and privkey.  Not recommended, but allowed.
#-----------------------------------------------------------------------------------------------------------------------
USM_USER  u1 - HMAC-MD5 6da6c69c64b7737360f8319d90e4d511         DES       959709e534eade82cecbacb42a10c90a          L -
USM_USER  u2 - HMAC-SHA f26562da268f21a916792d3f45500cd5a8163071 DES       3b3249ad3eb10d46d2731ee6fbaf8591          L -
USM_USER  u3 - HMAC-MD5 d1f86e9c9346253c10a4cd2da339b1db         DES       cfccde3249ba521ae4da0ddfc2b76ee7          N -
USM_USER  u4 - HMAC-SHA 42529b3c6c138c173e70db1050de8d74c04205cb DES       ef70a0a98d399a9189f3169e82010f3b46e694e2  N -
USM_USER  u5 - HMAC-MD5 2b0e2c55d452b5ada056d50e8a66ea35         DES       2b0e2c55d452b5ada056d50e8a66ea35          L -
USM_USER  u6 - HMAC-SHA aaa2f3b36f840549b6e8916b7b90430765dd3858 DES       aaa2f3b36f840549b6e8916b7b904307          L -
USM_USER  u7 - HMAC-MD5 5fbd3ad2fa6569d6c1e9ab4b83728b87         AESCFB128 bf686267600ff8f4b1354b857d186b55          L -
#-----------------------------------------------------------------------------------------------------------------------
# VACM_GROUP entries
# Format is:
#  groupName securityModel securityName storageType
#-----------------------------------------------------------------------------------------------------------------------
VACM_GROUP group1 USM     u1 -
VACM_GROUP group1 USM     u2 -
VACM_GROUP group1 USM     u3 -
VACM_GROUP group1 USM     u4 -
VACM_GROUP group2 USM     u5 -
VACM_GROUP group2 USM     u6 -
VACM_GROUP group2 USM     u7 -
VACM_GROUP group3 SNMPv1  publicv1  -
VACM_GROUP group3 SNMPv2c publicv2c -
VACM_GROUP group4 SNMPv1  MVSsubagent  -
VACM_GROUP group4 SNMPv2c MVSsubagent  -
VACM_GROUP group5 SNMPv1  scsecnamev1      -
VACM_GROUP group5 SNMPv2c scsecnamev2c     -
VACM_GROUP group5 SNMPv2c scsecnameIPv6v2c -
VACM_GROUP group6 SNMPv1  publicIPv6v1 -
VACM_GROUP group6 SNMPv2c publicIPv6v2c -
#------------------------------------------------------------------------------------------------------------------
# VACM_VIEW entries
# Format is:
#  viewName viewSubtree viewMask viewType storageType
#------------------------------------------------------------------------------------------------------------------
VACM_VIEW bigView        internet   - included -
VACM_VIEW prettyBigView  internet   - included -
VACM_VIEW prettyBigView  interfaces - excluded -
VACM_VIEW mediumView     system     - included -
VACM_VIEW mediumView     interfaces - included -
VACM_VIEW mediumView     tcp        - included -
VACM_VIEW mediumView     udp        - included -
VACM_VIEW mediumView     icmp       - included -
VACM_VIEW smallView      snmp       - included -
VACM_VIEW subagentView   dpiPort    - included -
#------------------------------------------------------------------------------------------------------------------
# VACM_ACCESS entries
# Format is:
#  groupName contextPrefix contextMatch securityLevel securityModel readView writeView notifyView storageType
#------------------------------------------------------------------------------------------------------------------
VACM_ACCESS  group1 - - AuthPriv     USM     bigView       bigView       bigView   -
VACM_ACCESS  group1 - - AuthNoPriv   USM     bigView       prettyBigView bigView   -
VACM_ACCESS  group1 - - noAuthNoPriv USM     smallView     smallView     smallView -
VACM_ACCESS  group2 - - AuthPriv     USM     bigView       bigView       bigView   -
VACM_ACCESS  group2 - - AuthNoPriv   USM     bigView       mediumView    smallView -
VACM_ACCESS  group2 - - noAuthNoPriv USM     bigView       mediumView    -         -
VACM_ACCESS  group3 - - noAuthNoPriv SNMPv1  mediumView    mediumView    mediumView -
VACM_ACCESS  group3 - - noAuthNoPriv SNMPv2c bigView       bigView       bigView   -
VACM_ACCESS  group4 - - noAuthNoPriv SNMPv1  subagentView  -             -         -
VACM_ACCESS  group4 - - noAuthNoPriv SNMPv2c subagentView  -             -         -
VACM_ACCESS  group5 - - noAuthNoPriv SNMPv1  mediumView    mediumView    mediumView -
VACM_ACCESS  group5 - - noAuthNoPriv SNMPv2c bigView       bigView       bigView   -
VACM_ACCESS  group6 - - noAuthNoPriv SNMPv1  bigView       bigView       bigView   -
VACM_ACCESS  group6 - - noAuthNoPriv SNMPv2c bigView       bigView       bigView   -
#------------------------------------------------------------------------------------------------------------------
# NOTIFY entries
# Format is:
#  notifyName tag type storageType
#------------------------------------------------------------------------------------------------------------------
NOTIFY notify1 traptag trap -
*NOTIFY notify2 informtag inform -
#------------------------------------------------------------------------------------------------------------------
# TARGET_ADDRESS
# Format is:
#  targetAddrName tDomain tAddress tagList targetParams timeout retryCount storageType tMask tMMS
#------------------------------------------------------------------------------------------------------------------
TARGET_ADDRESS Target1 UDP  9.67.113.10      traptag   trapparms1   - - - -      -
TARGET_ADDRESS Target2 UDP  9.67.113.5:2162  traptag   trapparms2   - - - -      -
TARGET_ADDRESS Target3 UDP  127.0.0.1        traptag   trapparms3   - - - -      -
*TARGET_ADDRESS Target4 UDP  127.0.0.1       informtag informparms  - - - -      -
TARGET_ADDRESS Target5 UDP6 ::1              traptag   trapparms2   - - - -      -
TARGET_ADDRESS Target6 UDP6 12AB::2          tagIPv6   comparmsIPv6 - - - 128..0  -
#------------------------------------------------------------------------------------------------------------------
# TARGET_PARAMETERS
# Format is:
#  paramsName mpModel securityModel securityName securityLevel storageType
#------------------------------------------------------------------------------------------------------------------
TARGET_PARAMETERS trapparms1   SNMPv1  SNMPv1  publicv1         noAuthNoPriv -
TARGET_PARAMETERS trapparms2   SNMPv2c SNMPv2c publicv2c        noAuthNoPriv -
TARGET_PARAMETERS trapparms3   SNMPv3  USM     u2               AuthNoPriv   -
*TARGET_PARAMETERS informparms SNMPv3  USM     u4               noAuthNoPriv -
TARGET_PARAMETERS comparmsIPv6 SNMPv2c SNMPv2c scsecnameIPv6v2c noAuthNoPriv -
#------------------------------------------------------------------------------------------------------------------
# NOTIFY_FILTER_PROFILE
# Format is:
#  targetParamsName profileName storageType
#------------------------------------------------------------------------------------------------------------------
NOTIFY_FILTER_PROFILE trapparms3  filProf    -
#------------------------------------------------------------------------------------------------------------------
# NOTIFY_FILTER
# Format is:
#  profileName filterSubtree filterMask filterType storageType
#------------------------------------------------------------------------------------------------------------------
NOTIFY_FILTER filProf  authenticationFailure - included -
#------------------------------------------------------------------------------------------------------------------
# COMMUNITY
# Format is:
#  communityName securityName securityLevel netAddr netMask storageType
# NOTE:
#  For CSV1R2 and later releases, the SNMP_COMMUNITY statement is recommended
#  rather than the COMMUNITY statement for community-based security.
#----------------------------------------------------------------------------------------------------------------------
COMMUNITY publicv1      publicv1      noAuthNoPriv   9.67.113.79 255.255.255.255 -
COMMUNITY publicv2c     publicv2c     noAuthNoPriv   0.0.0.0     0.0.0.0       -
COMMUNITY publicIPv6v1  publicIPv6v1  noAuthNoPriv   12ab::0     16            -
COMMUNITY publicIPv6v2c publicIPv6v2c noAuthNoPriv   0::0        0             -
COMMUNITY MVSsubagent   MVSsubagent   noAuthNoPriv   9.0.0.0     255.0.0.0       -
#----------------------------------------------------------------------------------------------------------------------
# SNMP_COMMUNITY
# Format is:
#  communityIndex communityName securityName contextEngineID contextName transportTag storageType
#----------------------------------------------------------------------------------------------------------------------
SNMP_COMMUNITY scindexv1      sccomnamev1       scsecnamev1      - - -       -
SNMP_COMMUNITY scindexv2c     sccomnamev2c      scsecnamev2c     - - -       -
SNMP_COMMUNITY scindexIPv6v2c sccomnameIPv6v2c  scsecnameIPv6v2c - - tagIPv6 -
#----------------------------------------------------------------------------------------------------------------------
# DEFAULT_SECURITY
# Format is:
#  securityPosture password privacy
#----------------------------------------------------------------------------------------------------------------------
DEFAULT_SECURITY semi-secure defaultpassword no
#----------------------------------------------------------------------------------------------------------------------
# Any SNMP agent configuration entries added by dynamic configuration
# (SET) requests get added to the end of the SNMPD.CONF file.
#----------------------------------------------------------------------------------------------------------------------
```

Figure 42. SNMPD.CONF sample

Migrating the PW.SRC file and SNMPTRAP.DEST file to the SNMPD.CONF file

If you want to continue to use community-based security (for SNMP protocols, SNMPv1 and SNMPv2c), but take advantage of some of the new SNMPv3 functions, or if you want to use the new SNMPV3 user-based security along with community-based security, you need to migrate your current configuration, defined in the PW.SRC and SNMPTRAP.DEST files, to the SNMPD.CONF format.

Steps for migrating the PW.SRC and SNMPTRAP.DEST files

This topic describes the steps of migrating the PW.SRC and SNMPTRAP.DEST files to the SNMPD.CONF file.

Procedure

Perform the following steps to migrate the PW.SRC and SNMPTRAP.DEST files to the SNMPD.CONF file:

1. For each community name defined in the the PW.SRC file, create an SNMP_COMMUNITY statement in the SNMPD.CONF file. This identifies the community names defined to the agent.

2. Create TARGET_ADDRESS entries to identify the address range permitted to use a particular community name. For each SNMP_COMMUNITY entry and for each range of addresses for which it is used, create a TARGET_ADDRESS entry. The TARGET_ADDRESS entries refer to related TARGET_PARAMETERS entries.

3. Create TARGET_PARAMETERS entries to identify the security model (SNMPv1 or SNMPv2c) to be used with the address on the corresponding TARGET_ADDRESS entries.

4. Define the following entries to determine which SNMP communities get access to which pieces of data and the type of access that they are allowed:

VACM_GROUP
: Specify one entry for each security model (in this case SNMPv1 or SNMPv2c) and use the community names from the PW.SRC file.

VACM_VIEW
: Specify one entry for each set of MIB object identifiers that you want to protect.

VACM_ACCESS
: Specify one entry that ties together the VACM_GROUP and VACM_VIEW entries and defines each group/view permission. You can define the group's permission to read, write, and receive notifications for the defined views.

---

5. To continue sending notifications, convert the entries in the SNMPTRAP.DEST file to entries in the SNMPD.CONF file.
   - Add one NOTIFY entry for type TRAP.
   - Add one TARGET_ADDRESS statement for each manager that receives a TRAP.
   - Optionally, configure a TARGET_PARAMETERS entry to identify the message model used in sending notifications to particular destinations. The default is SNMPv1, or specify SNMPv2c. Encryption and authentication are not used.

Results

Tip: For more detailed information about migrating z/OS SNMP configuration files from SNMPv1 and SNMPv2c to SNMPv3, see Technote Migrating z/OS SNMP to SNMPv3 at http://www-01.ibm.com/support/docview.wss?uid=swg27004972.

Example

For an example of using SNMPD.CONF statements to configure community-based security, see Figure 41 on page 1254.

SNMPD.BOOTS statement syntax

The syntax is:

engineID engineBoots

where:

engineID
: A string of 2–64 (must be an even number) hexadecimal digits. The engine identifier uniquely identifies the agent within an administrative domain. By default, the engine identifier is created using a vendor-specific formula and incorporates the IP address of the agent. However, a customer can choose to use any engine identifier that is consistent with the snmpEngineID definition in RFC 3411 and that is also unique within the administrative domain.

engineBoots
: The number of times (in decimal) the agent has been restarted since the engineID was last changed.

Note:
1. *engineID* and *engineBoots* must be specified in order and on the same line.
2. Comments are specified in the file by starting the line with either an asterisk (*) or a # character.
3. No comments are allowed between the *engineID* and *engineBoots* values.
4. Only the first non-comment line is read. Subsequent lines are ignored.
5. If an error is detected in processing an entry and no appropriate default value can be assumed, the entry is discarded and an error message is written to the syslog daemon.

SNMPD.BOOTS search order

The search order for accessing SNMPD.BOOTS information is as follows. The first file found in the search order is used.
1. The name of a z/OS UNIX file or an MVS data set specified by the SNMPD_BOOTS environment variable.
2. /etc/snmpd.boots

Guideline: If the SNMPD.BOOTS file is not provided, the SNMP agent creates the file. If multiple SNMPv3 agents are running on the same MVS image, use the environment variable to specify different SNMPD.BOOTS files for the different agents. For security reasons, ensure unique engineIDs are used for different SNMP agents.

SNMP query engine (SNMPQE)

This topic describes the SNMP query engine (SNMPQE).

SNMP query engine cataloged procedure (SNMPPROC)

This topic shows the SNMPPROC cataloged procedure.

```
//SNMPQE   PROC MODULE=SQESERV,PARMS=''
//*
//* z/OS Communications Server
//* SMP/E Distribution Name: EZAEB01W
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5694-A01
//*               (C) Copyright IBM Corp. 1989, 2002
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R4
//*
//SNMPQE   EXEC PGM=&MODULE,PARM='&PARMS',
//             REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here.  If you add
//*        them to STEPLIB, they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZADSIL,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//*
//*        The SYSMDUMP DD statement will cause MVS to provide
//*        an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
```

```
//*
//*       MSSNMPMS identifies an optional data set for NLS support.
//*       It specifies the SNMP message repository.
//*
//*MSSNMPMS DD DSN=TCPIP.SEZAINST(MSSNMP),DISP=SHR
//*
//*       SYSTCPD explicitly identifies which data set is to be
//*       used to obtain the parameters defined by TCPIP.DATA
//*       when no GLOBALTCPIPDATA statement is configured.
//*       See the IP Configuration Guide for information on
//*       the TCPIP.DATA search order.
//*       The data set can be any sequential data set or a member of
//*       a partitioned data set (PDS).
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Figure 43. SNMP query engine cataloged procedure (SNMPPROC)

Specifying the SNMPQE parameters

The SQESERV module can be configured to start without parameters or you can add any of the following parameters to PARMS=' in the PROC statement of the SNMPQE cataloged procedure. For example,

```
//SNMPQE   PROC MODULE=SNMPQE,PARMS='-h MVSA'
```

Note:

1. These parameters are also valid when starting SNMPQE with the START command.
2. The commands are case sensitive. They must be entered in lowercase.
3. When starting SNMPQE in batch, do not use the 'POSIX(ON)' parameter. This alters the search order the query engine uses to find the configuration files, which could prevent it from locating any configuration files not explicitly pointed to using a standard environment variable. It can also have other unexpected results.

Parameter
Description

-d *trace_level*
Specifies the level of tracing to be run. Valid values for the trace level are:

0	No tracing (default)
1	Displays errors
2	In addition to errors, also displays SNMP query engine protocol packets sent and received
3	In addition to 2, also displays the SNMP packets sent and received
4	In addition to 3, also displays the buffers in hexadecimal format

-h *host_name*
Specifies the IP address to which to bind, so that SQESERV accepts connections only through that IP address. This parameter is useful if multiple IP addresses exist for a single host, and you want to restrict access from one side.

-it
Specifies that a trace of IUCV communication be done. This is only used for debugging the socket layer in a user’s application. It can result in a large amount of STDOUT output.

-tp port_number
Specifies the port at which the SNMP Query Engine listens for traps. If this option is not specified, the SNMP Query Engine listens on the well-known port 162. The valid values are 1 - 65 535.

See z/OS Communications Server: IP Diagnosis Guide for more information about tracing.

SNMP parameter data set (SNMPARMS) sample

Following is the SNMPARMS sample:

```
*--------------------------------------------------------------*
* Member name:                                                 *
*      SNMPARMS                                                *
*                                                              *
*                                                              *
* Copyright:    Licensed Materials - Property of IBM           *
*                                                              *
*              "Restricted Materials of IBM"                   *
*                                                              *
*              5647-A01                                        *
*                                                              *
*              (C) Copyright IBM Corp. 1977, 1998              *
*                                                              *
*              US Government Users Restricted Rights -         *
*              Use, duplication or disclosure restricted by    *
*              GSA ADP Schedule Contract with IBM Corp.        *
*                                                              *
* Status:      CSV2R6                                          *
*                                                              *
*                                                              *
* Function:                                                    *
*      Externalized paramters for SNMPIUCV module, the task    *
*      that communicates with the SNMP Query Engine.           *
*                                                              *
* Attributes:                                                  *
*      Read by DSIDKS to obtain actual parameters.             *
*                                                              *
* Library:                                                     *
*      On MVS:   Member SNMPARMS in NetView's DSIPRM dataset.  *
*      On VM:    File SNMPARMS NCCFLST on a NetView minidisk.  *
*                                                              *
* Change activity:                                             *
*                                                              *
*   PTM      DATE    DESCRIPTION                               *
* -------  -------   ------------------------------------------*
*          24Sep89   Initial version                           *
*          09Nov89   Final version                             *
*          20Feb90   Adapt defaults as recommended during test*
*          27Mar90   Adapt defaults for new AF_IUCV sockets    *
*          09Jul90   Remove not needed paramters               *
*                                                              *
*--------------------------------------------------------------*
*
  SNMPQE   SNMPQE   * Userid of SNMP Query Engine
  SNMPQERT 60       * Retry timer (seconds) for IUCV CONNECT
  SNMPRCNT 2        * Retry count for sending SNMP requests
  SNMPRITO 10       * Retry initial timeout (10ths of a second)
  SNMPRETO 2        * Retry backoff exponent (1=linear, 2=exponential)
  SNMPMMLL 80       * Line length for Multiline Messages 38/44
```

Figure 44. SNMP parameter data set (SNMPARMS) sample

Specifying the SNMPARMS parameters

You can change the following parameters in SNMPARMS:

Parameter
: **Description**

SNMPQE *name*
: The name of the SNMP query engine started procedure. This value is case sensitive. The default address space name is SNMPQE. If you change the name of the SNMP query engine started procedure, you must change this parameter to match the new procedure name.

SNMPQERT *seconds*
: The retry timer, in seconds, for IUCV CONNECT. When SNMPIUCV is started, it tries to connect to the SNMP query engine. If the connection fails or breaks, SNMPIUCV retries a connect every *n* seconds, as specified by this parameter. The valid range of values is 0 - 9999. The default is 60 seconds.

SNMPRCNT *number*
: The retry count for sending SNMP requests. This is the number of times the SNMP query engine resends an SNMP PDU when no response was received. If no response is received after all retries have been exhausted, the SNMP query engine returns a no response error for the SNMP request. The valid range of values is 0 - 255. The default is 2.

  If the request being sent by the SNMP query engine contains a community name that is not valid, no response is received. This causes the SNMP query engine to resend the request until the retry count is exhausted. If authentication failure traps are enabled, the agent generates multiple ***authentication-Failure*** traps, one for the initial request and one for each of the retries.

SNMPMMLL *length*
: The line length for multiline messages 38 and 44. The maximum length is 255. A value of 80 allows the complete text to appear on an 80-character-wide screen. The default and minimum acceptable line length value is 80.

SNMPRETO *exp*
: The retry back-off exponent. Specifies whether the timeout value between retries of an SNMP request is calculated linearly or exponentially. The valid values are 1 (linear) or 2 (exponential). The default is 2.

  For example, if the retry timeout was 1 second, SNMPRETO of 1 causes a new retry to be sent at constant 1-second intervals until all retries have been sent. SNMPRETO of 2 causes the first retry to be sent after 1 second, the second retry 2 seconds later, the third retry 4 seconds later, and so on until all retries have been sent.

SNMPRITO *tenths_seconds*
: The timeout value for a request specified in tenths of a second. After sending an SNMP request to an agent, the SNMP query engine waits the specified number of tenths of a second for a response.

  - If the retry count (SNMPRCNT) is greater than 0, the SNMP request is sent again if a response is not received in this time.
  - If the retry count (SNMPRCNT) is 0, a `no response` error is sent to the NetView program, if a response is not received within this period of time.

The valid range of values is 0 - 255. The default is 10 tenths of a second.

MIBDESC.DATA statement

The MIBDESC.DATA statement syntax is:

```
short_name asn.1_name type time_to_live
```

where:

- *short name* is the textual name for the MIB object, either as defined in the MIB definition or chosen by the customer.
- *asn.1_name* is the MIB object identifier that describes the location of the object in the MIB tree.
- *type* is the syntax of the MIB object.
- *time_to_live* is the number of seconds the SNMP Query Engine can cache the object before requesting an updated copy from the SNMP agent.

The following SNMP variable type values (from SMI version 1) are supported in the type field of the MIBDESC.DATA statement:

- Number for integers
- String for octet strings
- Object for object identifiers
- Internet for IP addresses
- Counter for counters (unsigned)
- Gauge for gauge (unsigned)
- Ticks for time ticks
- Display for display strings
- Table for table header variables
- Empty for no value

MIBDESC.DATA search order

The search order for accessing MIBDESC.DATA is as follows. The first file found in the search order is used.

1. The name of a z/OS UNIX file or an MVS data set specified by the MIB_DESC environment variable.
2. *hlq*.MIBDESC.DATA, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file being used .

MIBDESC environment variables

Table 93 provides a list of environment variables used by MIBDESC that can be tailored to a particular installation:

Table 93. MIBDESC environment variables

Environment variable	**Server, Client or Command-type application**	**Description**	**Any specific coding rules (or a link to syntax)**
MIB_DESC	SNMP Query Engine	Specifies the location of the MIBDESC.DATA configuration file	None

z/OS UNIX snmp command

This topic describes the z/OS UNIX snmp command.

Environment variables

Table 94 provides a list of environment variables used by the command that you can modify to use with your installation environment:

Table 94. z/OS UNIX snmp command environment variables

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules (or a link to syntax)
MIBS_DATA	z/OS UNIX snmp command	Specifies the location of the MIBS.DATA configuration file.	None
OSNMP_CONF	z/OS UNIX snmp command	Specifies the location of the OSNMP.CONF configuration file.	None

OSNMP.CONF search order

The following search order for this file enables different copies of the file to be used by different users:

1. A z/OS UNIX file or MVS data set pointed to by the OSNMP_CONF environment variable
2. /etc/osnmp.conf
3. /etc/snmpv2.conf

OSNMP.CONF statement syntax

The configuration file is required when sending requests to the SNMPv2 or SNMPv3 nodes in your network. The configuration file can also be used to send SNMPv1 requests.

The syntax of a statement in the configuration file is:

```
winSNMPname targetAgent admin secName password context secLevel
authProto authKey privProto privKey NOSVIPA
```

Field definitions

winSNMPname
An administrative name that the snmp command uses to locate an entry in the configuration file. There is no default value. This field is specified on the -h option (maximum 32 characters).

targetAgent
Host name or IP address (IPv4 dotted decimal or IPv6 colon hexadecimal) of the node of the target agent (maximum 80 characters). There is no default value. To direct the command to a port other than 161, specify *host..port#* (with two periods between the host and port number). For example, for port 222 at mvs150, specify mvs150..222. Port number, if specified, must be in the range of 1 to 65535. If the host is specified by a host name or an IPv4 dotted decimal address and a port number is also specified, a colon (:) can be used to separate the two values instead of two periods.

admin Specifies the administrative model supported by the targetAgent. Valid values are:

- snmpv1 - Community-based SNMPV1 security

- snmpv2c - Community-based SNMPV2 security
- snmpv3 - User-based SNMPV3 security

There is no default value.

secName
Specifies the security name of the principal using this configuration file entry. For user-based security, this is the userName. The user must be defined at the targetAgent. This field is ignored unless snmpv3 is specified for the admin keyword. A valid value is a user name of 1–32 characters. There is no default.

password
Specifies the password to be used in generating the authentication and privacy keys for this user. If a password is specified, it is used to automatically generate any needed keys and the "authKey" and "privKey" fields below are ignored. This field is ignored unless snmpv3 is specified for the admin keyword. If no password is desired, set field to a single dash (-). (The minimum is eight characters, and the maximum is 64 characters.)

Guideline: You should not use the password instead of keys in this configuration file, because using keys is more secure than storing passwords in this file.

context
The SNMP contextName to be used at the target agent. The contextName is needed only at agents that support multiple contexts; otherwise, the only context supported is the null context, which is the default value of this keyword. The z/OS Communications Server SNMP agent does not support multiple contexts. This field is ignored unless snmpv3 is specified for the admin keyword. If the blank "" context selector is desired, set this field to a single dash (-). (The maximum is 32 characters).

secLevel
Specifies the security level to be used in communicating with the target SNMP agent when this entry is used. This field is ignored unless snmpv3 is specified for the admin keyword. Valid values are noAuthNoPriv or none to indicate that no authentication or privacy is requested; AuthNoPriv or auth to indicate that authentication is requested but privacy is not requested; AuthPriv or priv to indicate that both authentication and privacy are requested; or a dash (-) to indicate the default value (noAuthNoPriv).

authProto
SNMP authentication protocol to be used in communicating with the target SNMP agent when this entry is used. This field is ignored unless snmpv3 is specified for the admin keyword. The following values are valid:

- HMAC-MD5
- HMAC-SHA
- dash (-). Indicates no authentication.

authKey
Specifies the SNMP authentication key to be used in communicating with the target SNMP agent when this entry is used. This key must be the nonlocalized key. This field is ignored if the password keyword is used. This field is ignored unless snmpv3 is specified for the admin keyword and a nondefault value is specified for authProto. Valid values are 16 bytes

(32 hex digits) when authProto is HMAC-MD5 and 20 bytes (40 hex digits) when authProto is HMAC-SHA. A dash (-) indicates the default value, which is no key.

privProto
Specifies the SNMP privacy protocol to be used in communicating with the target SNMP agent when this entry is used. This field is ignored unless snmpv3 is specified for the admin keyword. The following values are valid:

DES
Indicates CBC-DES.

AESCFB128
Indicates AES 128-bit CFB mode.

dash (-)
Indicates the default value, which is no privacy.

Requirement: For the AES privacy protocol, ICSF must be active. For detailed information about configuring ICSF, see z/OS Cryptographic Services ICSF Administrator's Guide.

privKey
Specifies the SNMP privacy key to be used in communicating with the target SNMP agent when this entry is used. This key must be the nonlocalized key. This field is ignored if the password keyword is used. The privacy and authentication keys are assumed to have been generated using the same authentication protocol (for example, both with HMAC-MD5 or both with HMAC-SHA). This field is ignored unless snmpv3 is specified for the admin keyword and a nondefault value is specified for privProto. Valid values are 16 bytes (32 hex digits) when authProto is HMAC-MD5, 20 bytes (40 hex digits) when authProto is HMAC-SHA, or a dash (-) to indicate the default value (no key).

NOSVIPA
The NOSVIPA keyword is an optional value. If specified, it indicates the osnmp command should cause physical interface addresses to be used as the originating address in packets sent by the osnmp command to this host. NOSVIPA is disabled by default, meaning that SOURCE VIPA addresses can be used. If specified, NOSVIPA must be either the fourth parameter (for community-based security) or the twelfth parameter (for user-based security).

Statement syntax rules

- All parameters for an entry must be contained on one line in the configuration file.
- A dash (-) indicates the default value for a keyword.
- Sequence numbers are not allowed on the statements.
- Comments begin with a # character in column 1.
- The secName and password parameters are case sensitive.
- The pwtokey command can be used to generate the authentication and privacy keys. For information about the pwtokey command, see z/OS Communications Server: IP System Administrator's Commands.
- Because the osnmp command supports both issuance of SNMP requests and receipt of SNMP traps, the entries in the OSNMP.CONF file must be defined for both uses. Multiple entries for the same USM user are allowed within the file.

This can be useful when defining different security levels for the same user. If multiple entries for the same USM user are defined, be aware that only the first one in the file can be used for receiving notifications. If multiple entries for the same USM user are defined and the user receives notifications, the definition with the highest (most stringent) securityLevel should be defined first. Doing so allows the user to be used for any level of security equal to or lower (less stringent) than the securityLevel defined.

Restriction: You cannot specify scope information about any values in the OSNMP.CONF file that represent IP addresses or host names.

OSNMP.CONF sample

Requirement: The osnmp command requires that all fields for a given entry are specified on a single line. For readability, the following sample has been formatted such that long entries are wrapped to the next line.

```
# osnmp.conf sample
#   (used as /etc/snmpv2.conf unless OSNMP_CONF environment variable set)
#
# Sample file showing format of configuration file for the osnmp command
#
# Licensed Materials - Property of IBM
# "Restricted Materials of IBM"
# 5650-ZOS
# Copyright IBM Corp. 1996, 2013
# Status = CSV2R1
# SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZASNV2C
#
#----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
#
# Format of entries (SNMPv1 and SNMPv2c):
#
#   winSnmpName  targetAgent  admin  nosvipa
#
# Format of entries (SNMPv3):
#
#   winSnmpName  targetAgent  admin  secName  password  context secLevel authProto  authKey  privProto  privKey
#
#----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
#----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Community-based security  (SNMPv1 and SNMPv2c)
#----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
v1       127.0.0.1   snmpv1
v2c      127.0.0.1   snmpv2c
v2c_ipv6 ::1         snmpv2c
mvs1     9.67.113.79 snmpv2c
# mvs2   mvs2c       snmpv2c    nosvipa
# mvs3   mvs3:1061   snmpv2c
mvs4     12ab::2     snmpv2c
#
#----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
# User-based Security Model (USM with SNMPV3)
#
# Notes
# - Keys in this file must not be localized.
# - All keys need to be regenerated using the pwtokey command in order
#   for these sample entries to actually be used.
# - In this sample:
#   - Keys are generated for use with engineID 00000002000000000943714F
#   - Authentication keys were generated with password of
#     username+"password", such as "u1password"
#   - Privacy keys were generated with password of
#     username+"privpass", such as "u1privpass"
#
# Format of entries (SNMPv3):
#
#   winSnmpName  targetAgent  admin  secName  password  context secLevel authProto  authKey  privProto  privKey
#----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
#
# v3mpk: SNMPv3 with HMAC-MD5, authentication and privacy, using keys
v3mpk   127.0.0.1 snmpv3 u1 -         - AuthPriv     HMAC-MD5 7a3e34265e0e029f27d8b4235ecfa987         DES eac02a0d9fe90eca7911fdcaba20deae
#
# v3mak: SNMPv3 with HMAC-MD5, authentication without privacy, using keys
v3mak   127.0.0.1 snmpv3 u1 -         - AuthNoPriv   HMAC-MD5 7a3e34265e0e029f27d8b4235ecfa987         -   -
#
# v3n:   SNMPv3 with no authentication or privacy
v3n     127.0.0.1 snmpv3 u1 -         - noAuthNoPriv -        -                                        -   -
#
# v3mpk_ipv6: SNMPv3 with HMAC-MD5, authentication and privacy, using keys, with IPv6 target host address
v3mpk_ipv6    ::1 snmpv3 u1 -         - AuthPriv     HMAC-MD5 7a3e34265e0e029f27d8b4235ecfa987         DES eac02a0d9fe90eca7911fdcaba20deae
#
# v3spk: SNMPv3 with HMAC-SHA, authentication and privacy, using keys
v3spk   127.0.0.1 snmpv3 u2 -         - AuthPriv     HMAC-SHA 76784e5935acd6033a855df1fac42acb187aa867 DES adaaf313277a55a3df3a8d2fb70192c427799e0c
#
# v3sak: SNMPv3 with HMAC-SHA, authentication without privacy, using keys
v3sak   127.0.0.1 snmpv3 u2 -         - AuthNoPriv   HMAC-SHA 76784e5935acd6033a855df1fac42acb187aa867 -   -
#
# v3mpk2: SNMPv3 with HMAC-MD5, authentication and privacy, using non-localized keys at agent
v3mpk2  127.0.0.1 snmpv3 u3 -         - AuthPriv     HMAC-MD5 d1f86e9c9346253c10a4cd2da339b1db         DES cfccde3249ba521ae4da0ddfc2b76ee7
#
# v3spk2: SNMPv3 with HMAC-SHA, authentication and privacy, using non-localized keys at agent
v3spk2  127.0.0.1 snmpv3 u4 -         - AuthPriv     HMAC-SHA 42529b3c6c138c173e70db1050de8d74c04205cb DES ef70a0a98d399a9189f3169e82010f3b46e694e2
```

```
#
# v3mpp: SNMPv3 with HMAC-MD5, authentication and privacy, using password to generate keys
v3mpp   127.0.0.1 snmpv3 u5 u5password - AuthPriv    HMAC-MD5 -                                        DES -      nosvipa
#
# v3map: SNMPv3 with HMAC-MD5, authentication without privacy, using password to generate keys
v3map   127.0.0.1 snmpv3 u5 u5password - AuthNoPriv  HMAC-MD5 -                                        -   -      nosvipa
#
# v3spp: SNMPv3 with HMAC-SHA, authentication and privacy, using password to generate keys
v3spp   127.0.0.1 snmpv3 u6 u6password - AuthPriv    HMAC-SHA -                                        DES -
#
# v3sap: SNMPv3 with HMAC-SHA, authentication without privacy, using password to generate keys
v3sap   127.0.0.1 snmpv3 u6 u6password - AuthNoPriv  HMAC-SHA -                                        -   -
#
# v3mpka: SNMPv3 with HMAC-MD5, authentication and privacy protocol AESCFB128, using keys
v3mpka  127.0.0.1 snmpv3 u7 -          - AuthPriv    HMAC-MD5 15549009e2401748e8077fa17bf64c9b         AESCFB128 90009683501c78a6f87575bdad5455bc
```

Figure 45. OSNMP.CONF sample

MIBS.DATA statement syntax

The MIBS.DATA statements can be used to specify character (usually called textual) names for MIB objects not defined in any compiled MIB supplied with z/OS Communications Server. You can then use these character/textual names as the name of the objects on the osnmp command.

The format of a statement in this file is:

character_object_name object_identifier object_type

Field definitions

character_object_name
The character or textual name of the MIB object. The *character_object_name* value can contain both uppercase and lowercase letters.

object_identifier
The ASN.1 value for the MIB object.

object_type
The SMI_type for the MIB object. The valid SMI_type values are:
- bitstring
- counter
- counter32
- counter64
- dateAndTime
- display or display string
- integer
- integer32
- ipaddress
- gauge
- gauge32
- nsapaddress
- null
- objectidentifier or OID
- octetstring
- opaque
- opaqueascii
- snmpAdminString
- timeticks

    - uinteger
- The maximum length of each statement in this file is 2048 bytes.
- All parameters for each character or textual name must be on the same statement.
- Sequence numbers are not allowed on the statements.
- Comments begin with a # character in column 1.

MIBS.DATA search order

The search order for accessing the MIBS.DATA information is as follows. The first file found in the search order is used.

- The name of a z/OS UNIX file or an MVS data set specified by the MIBS_DATA environment variable
- /etc/mibs.data z/OS UNIX file

TRAPFWD daemon

The TRAPFWD daemon forwards traps from the SNMP agent to network management applications. It listens for traps on port 162 and forwards them to all configured managers.

Starting TRAPFWD from an MVS console

Update cataloged procedure TRAPFWD by copying the sample in SEZAINST(TRAPFWD) to your system.

The following is a sample JCL Procedure for starting TRAPFWD from MVS:

```
//TRAPFWD   PROC
//*
//* Sample procedure for running the z/OS UNIX Trap Forwarder daemon
//*
//* z/OS Communications Server Version 1 Release 7
//* SMP/E Distribution Name: SEZAINST(EZASNTPR)
//*
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               5694-A01
//*               (C) Copyright IBM Corp. 2000, 2005
//*
//* Status:       CSV1R7
//*
//TRAPFWD EXEC PGM=EZASNTRA,REGION=4096K,TIME=NOLIMIT,
//   PARM='POSIX(ON) ALL31(ON)/-d 0'
//*
//*** Notes:
//*
//* - The C runtime libraries should be in the system's link list
//*   or this sample procedure will need to STEPLIB to them.
//*
//* - TCP/IP runtime libraries should also be in the system's link
//*   list.
//*
//* - TRAPFWD must find the name (TCPIPJOBNAME in TCPIP.DATA) that
//*   it should be associated with. The OE function __iptcpn() is
//*   used to find this name. It is suggested that the parmlist
//*   be modified to set the environment variable
//*   RESOLVER_CONFIG to point to the correct resolver file when
//*   multiple INET Physical File Systems are started.
//*
//*   If only one INET PFS will be started then /etc/resolv.conf
//*   may be used.
//*
//* - The TRAPFWD daemon can also be invoked from the OMVS shell as
//*   a shell command.
//*
//*
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*
```

Figure 46. TRAPFWD cataloged procedure

Specifying TRAPFWD parameters

The following parameters are available for TRAPFWD:

Parameter
Description

-d *n*
The -d flag indicates the level of debug information that is desired. The valid values are:

- 0 - No tracing
- 1 - Minimal tracing. Ttrace address from which the trap is received.
- 2 - In addition to 1, trace addresses to which the trap packet is forwarded.
- 3 - In addition to 2, trace trap packets.

If the -d parameter is not specified, the default value of 0 is used.

-p *port_number*
: The -p flag indicates the UDP port at which the daemon should listen for traps. The default is UDP port 162.

-l *max_packet_len*
: The -l flag indicates the maximum packet length of the trap datagram that has to be forwarded. The valid values are 4096 (4K) to 16384 (16K). The default value is 4096. Note that if the ADD_RECVFROM_INFO option is specified, then the maximum packet size is be max_packet_len minus the length of the address information.

-?
: The -? flag displays the usage statement for the trap forwarder daemon. If the -? option is specified, all the other options are ignored.

TRAPFWD environment variables

Table 95 provides a list of environment variables used by TRAPFWD that can be tailored to a particular installation:

Table 95. TRAPFWD environment variables

Environment variable	Server, Client or Command-type application	Description	Any specific coding rules (or a link to syntax)
TRAPFWD_CONF	TRAPFWD daemon	Specifies the location of the TRAPFWD.CONF configuration file	None

Starting TRAPFWD from the UNIX shell

The trapfwd command is used to start the trap forwarder daemon.

To start TRAPFWD from the UNIX shell:

```
                    ┌─-d ──0─┐   ┌─-p ──162──────────┐   ┌─-l ──4096───────────┐
►►──trapfwd──┬──┼────────┼───┼───────────────────┼───┼─────────────────────┼──┬──────────►◄
             │  └─-d ──n─┘   └─-p──port_number ──┘   └─-l ──max_packet_len─┘  │
             └─?───────────────────────────────────────────────────────────────┘
```

TRAPFWD.CONF statement syntax

The format of a statement in this file is:

`host_name port_number number`

Field definitions

host_name
: The host name or IP address (IPv4 dotted decimal or IPv6 colon hexadecimal) to which the trap should be forwarded. If a dash (-) is used, then the local host name is used.

  Restriction: Scope information cannot be specified for the *host_name* value.

port_number
: The port number to which the trap should be forwarded. There is no default value.

number
: This field is optional. If a value of ADD_RECVFROM_INFO is specified,

the received from information is appended to the trap. The default is not to append the received from information.

Usage notes

Lines starting with an asterisk (*) or a # character are considered comment lines.

TRAPFWD.CONF search order

The following search order is used to access the TRAPFWD.CONF information:

1. A z/OS UNIX file or an MVS data set specified by the TRAPFWD_CONF environment variable
2. /etc/trapfwd.conf

The first file found in the search order is used.

If the environment variable is set and if the file specified by the environment variable is not found, the Trap Forwarder daemon terminates.

TRAPFWD examples

To start the trap forwarder daemon on the standard port (port 162), enter:

```
# trapfwd
```

To start the trap forwarder daemon on a particular port (port 5062), enter:

```
# trapfwd -p 5062
```

Chapter 25. Remote print server

This topic contains the following information:

LPD server cataloged procedure (LPSPROC)

The following sample shows the Remote print server (LPD) server cataloged procedure (LPSPROC).

```
//LPSERVE  PROC MODULE=LPD,
//         LPDDATA=TCPIP.SEZAINST(LPDDATA),
//         LPDPRFX='PREFIX TCPIP',
//         DIAG=''
//*
//*        z/OS Communications Server
//*        SMP/E Name: EZAEB019 alias LPSPROC in library SEZAINST
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5694-A01
//*               Copyright IBM Corp. 1996, 2008
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R10
//LPD   EXEC PGM=MVPMAIN,
//  PARM=('&MODULE,ERRFILE(SYSERR),HEAP(512)',
//       'NOSPIE/ ''&LPDDATA'' &LPDPRFX &DIAG'),
//       REGION=6M,TIME=1440
//SPOOL OUTPUT CHARS=GT12
//STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR
//LPD1 OUTPUT CHARS=GT12
//*
//*        SYSPRINT contains runtime diagnostics from LPD. It
//*        can be a data set or SYSOUT.
//*
//SYSPRINT DD SYSOUT=*
//*
//*        SYSERR contains runtime diagnostics from Pascal. It can be
//*        a data set or SYSOUT.
//*
//SYSERR   DD SYSOUT=*
//*
//*        SYSDEBUG receives output that is generated when the TRACE
//*        parameter is specified in the PARM on the EXEC card.
//*        It can be a data set or SYSOUT.
//*
//SYSDEBUG DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//*        The SYSMDUMP DD statement will cause MVS to provide
//*        an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA
//*        when no GLOBALTCPIPDATA statement is configured.
//*        See the IP Configuration Guide for information on
//*        the TCPIP.DATA search order.
//*        The data set can be any sequential data set or a member of
//*        a partitioned data set (PDS).
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.SEZAINST(TCPDATA)
```

Figure 47. LPD Server cataloged procedure (LPSPROC)

Sample LPD server configuration data set (LPDDATA)

The following sample shows the LPD server configuration data set (LPDDATA).

```
;LPD CONFIGURATION DATA SET
;=========================
;
```

```
; COPYRIGHT = NONE.
; Change Activity
; $L1=MV11199 HTP320 951206 RTPMCL: Added comments
; $01=PN66730 HTP310 960131 RTPMCL: Added change flag
;    =MV14695 TCPV34 970910 YANG  : Removed sequence numbers
;    =MV15117 TCPV34 970929 AGIUS : Added OBEY statement
;
; This data set describes the printers and punches (which are both
; called SERVICE) that are usable from LPR client programs for this
; host.
;
; Each SERVICE must be described as LOCAL, NJE, or REMOTE.  Data for
; LOCAL services are managed directly by JES.  Data for NJE services
; are managed by NJE.  REMOTE services' data are forwarded to another
; LPD (print server).
;
; You can control which types of printing or punching can be done
; through a particular SERVICE with FILTERS.  The 4 currently
; available FILTERS are:
;
;       f    which paginates the data set at the size of the page given.
;            It also truncates lines if they exceed a maximum length.
;       l    which does not insert pagination but will truncate lines
;            as the "f" filter does.
;       p    which paginates the data set, adding titles, the date, and
;            page numbers as well as providing line truncation.
;       r    which prints the data set, interpreting the first column
;            of each line as FORTRAN carriage control.
;
;
; Most printer SERVICEs should allow all three but you probably only
; want to specify "l" for punches.
;
; The LINESIZE option can be used to limit the length of lines written
; by the filters.
;
; The PAGESIZE option can be used for filters that do pagination to
; specify how many lines should appear on a page.
;
; The RACF option will cause the server to verify that a user knows
; the account password for a user ID on this host.
;
; These statements define a LOCAL PRINTER SERVICE called locprt1, which
; is a conventional printer that will use the JES printing facilities.
;
;DEBUG
 SERVICE locprt1 PRINTER
    LOCAL
    FILTERS f l p r
    LINESIZE 132
    PAGESIZE 60
;
; These statements define an NJE PRINTER SERVICE called njeprt1, which
; provides access to the NJE service on this system.
;
;SERVICE njeprt1 PRINTER
;   NJE DEST=RALVMM IDENTIFIER=JOHN OUTPUT=LPD1
;   FILTERS f l p r
;   LINESIZE 132
;   PAGESIZE 60
;
; These statements define a REMOTE SERVICE called pebprt, which
; provides access to the printing queues on another system.
; From an LPR client, specify the printer name defined on the SERVICE
; statement and the hostname or IP address of the host that this LPD
; is running on, NOT the names on the REMOTE statement.
;     Example: LPR fn (p pebprt h LPDSrvHostName
; The above is required if you wish to send the data to the REMOTE
; printer via this LPD.
;
;SERVICE pebprt PRINTER
;  REMOTE lpt1@PEBBLES.TCP.RALEIGH.IBM.COM
; FAILEDJOB MAIL
;
; These statements define a PUNCH SERVICE called pun1, which
; provides access to the JES controlled PUNCH.
;
;SERVICE pun1 PUNCH
;   LOCAL
;   FILTERS l
;   LINESIZE 80
;
; This statement specifys user IDs authorized to use the SMSG
; interface provided with LPD server.  The syntax is "OBEY user_id",
; where "user_id" is the list of authorized user IDs.  The following
; example allows three test user IDs to use the SMSG interface:
;
;OBEY TESTER01 TESTER02 TESTER03
;
```

Figure 48. Sample LPD server configuration data set (LPDDATA)

Specifying LPD server parameters

The system parameters required by the LPD server are passed by the PARM option on the EXEC statement of the LPD cataloged procedure. Update the following parameters as required.

LPDDATA='*data_set_name***'**
: Specifies the fully qualified name of the data set containing the LPD configuration statements.

  Guideline: This data set can be sequential or a member of a PDS.

LPDPRFX='PREFIX *your_prefix***'**
: Specifies the high-level qualifier to be used for temporary data sets created by the LPD server. Include both the PREFIX keyword and your qualifier in the quoted string. The qualifier can be up to 26 characters. If it is blank, it defaults to the procedure name. The LPD task requires the authority to create and modify data sets with this prefix.

DIAG='*options***'**
: Specifies any of the following diagnostic options in a quoted string of keywords separated by blanks. For example, `DIAG='VERSION TRACE'`

  VERSION
  : Displays the version number.

  TYPE
  : Activates high-level trace facility in the LPD server. Significant events, such as the receipt of a job for printing, are recorded in the //SYSOUT DD data set specified in your LPD server cataloged procedure.

  TRACE
  : Causes a detailed trace of activities within the LPD server to record in the //SYSOUT DD data set specified in your LPD server cataloged procedure.

    Tip: The detailed tracing can also be activated with the DEBUG statement in the LPD server configuration data set and with the TRACE command of the SMSG interface.

Restriction: The JCL PARM= statement has a limit of 100 characters.

Summary of LPD server configuration statements

The valid statements for this data set are listed in the following table.

Table 96. Summary of LPD server configuration statements

Statement	Description	See
DEBUG	Turns on tracing of all LPD processes.	"DEBUG statement"
JOBPACING	Restricts parallel processing of jobs to conserve memory.	"JOBPACING statement" on page 1296
OBEY	Specifies users IDs that can use the SMSG interface.	"OBEY statement" on page 1296
SERVICE	Specifies the name and Type of Service.	"SERVICE statement" on page 1297
STEPLIMIT	Restricts complexity of jobs received to conserve memory.	"STEPLIMIT statement" on page 1306
UNIT	Specifies type of DASD that LPD should use for temporary data sets.	"UNIT statement" on page 1307
VOLUME	Specifies the volume that LPD should use for temporary data sets.	"VOLUME statement" on page 1307

LPD server configuration data set statements

This topic includes the syntax rules and alphabetically listed definitions of the statements for the data set used to configure the LPD Server.

Syntax rules

In the LPD server configuration data set, tokens are delimited by blanks and record boundaries. All characters to the right of and including a semicolon are treated as comments.

DEBUG statement

Use the DEBUG statement to activate full tracing of the processing within the LPD server.

Syntax

```
►►──DEBUG──────────────────────────────────────────────────────────►◄
```

Parameters

There are no parameters for this statement.

Usage notes

- Detailed tracing can also be activated using the TRACE parameter on the PROC statement of the LPD server procedure or by specifying TRACE ON with the SMSG interface. The DEBUG statement can be placed anywhere in the data set but only affects those services following it. Including DEBUG as the first statement in the configuration data set allows trace messages to be written from the point LPD is initialized.
- LPD generates minimal tracing under the following conditions:

  - No value in DIAG parameter
  - TRACE not passed as a parameter
  - DEBUG not defined in the LPD configuration file
- Coding LPD with DIAG=Version results in minimal tracing plus the message EZB0614I. Coding LPD with DIAG=Type results in minimal tracing plus brief messages describing JOB status, such as:
  - JOBreceived
  - JobStartPRINTING
  - JOBcontinuePRINTING
  - JOBfinishPRINTING

  Coding LPD with DIAG=Trace results in configuration messages and details of print job.
- TRACE passed as a parameter yields the same results coding DIAG=TYPE.

JOBPACING statement

Use the JOBPACING statement to limit the number of jobs that the LPD server concurrently writes to the JES spool or send to another LPD server. This limits memory requirements in LPD, but does not cause any jobs to be lost. Received jobs are queued until they can be processed.

Syntax

```
                 ┌───────────┐
                 ▼           │
►►──JOBPACING────────limit───┴──────────────────────────────────────────►◄
```

Parameters

limit
: An integer specifying the maximum number of jobs that the LPD server concurrently writes to the JES spool or send to another LPD server.

Usage notes

- Concurrent processing of jobs requires memory for control blocks and large I/O buffers. Some concurrent job processing keeps a long job or slow receiving LPD from delaying all the other jobs. Too much concurrent processing causes thrashing and requires extensive memory.
- JOBPACING defaults to the preferred value 5 when the keyword is not specified. Increasing this value might cause memory allocation problems with certain system configurations.
- If LPD runs out of memory, reduce the value of either JOBPACING or STEPLIMIT.

OBEY statement

Use the OBEY statement to specify user IDs authorized to use the SMSG interface provided with LPD server.

Syntax

```
            ┌──────────────┐
            ▼              │
►►──OBEY────────user_id────┴──────────────────────────────────────────────►◄
```

Parameters

user_id
: The user IDs authorized to use the SMSG interface. See z/OS Communications Server: IP Configuration Guide for more information.

Examples

Code the following statement to allow three test user IDs to use the SMSG interface:

```
OBEY TESTER01 TESTER02 TESTER03
```

Usage notes

Multiple user IDs can be specified on the OBEY statement. More than one OBEY statement can be included in the data set.

SERVICE statement

Use the SERVICE statement to specify the name and Type of Service for the printers and punches used by the LPD server. This service name is used in the LPR command.

Requirement: The parameters shown on separate lines must be coded on separate lines. Follow the example in the sample configuration data set shown in “Sample LPD server configuration data set (LPDDATA)” on page 1292.

Syntax

```
►►──SERVICE──name──┬─PRINTER─┬──┬─LOCAL──local_options──────┬─────────────►
                   ├─RECFMU──┤  ├─NJE──nje_options──────────┤
                   ├─RECFMUA─┤  └─REMOTE──┬─printer@host─┬──┘
                   ├─PUNCH───┤            └─EPORT────────┘
                   └─NONE────┘
```

```
   ┌─────────────────────────────────────────────────────────────────┐
   ▼                                                                 │
►──┬─────────────────────────────────────────────────────────────┬──┴─────────►◄
   ├─EXIT───┬─START─┬──program──────────────────────────────────┤
   │        └─END───┘                                            │
   ├─FAILEDJOB───┬─MAIL────┬─────────────────────────────────────┤
   │             └─DISCARD─┘                                     │
   │              ┌─────────┐                                    │
   │              ▼         │                                    │
   ├─FILTERS───┬─f────┬─────┴────────────────────────────────────┤
   │           ├─l────┤                                          │
   │           ├─lASf─┤                                          │
   │           ├─p────┤                                          │
   │           └─r────┘                                          │
   │ ┌─LINESIZE 132──────┐                                       │
   ├─┴─LINESIZE ──length─┴───────────────────────────────────────┤
   ├─NLSTRANSLATE──... see separate diagram ...──────────────────┤
   │ ┌─PAGESIZE 60──────┐                                        │
   ├─┴─PAGESIZE ──lines─┴────────────────────────────────────────┤
   ├─RACF────────────────────────────────────────────────────────┤
   │ ┌─SMTP SMTP─────────────────────────────────────────────┐   │
   ├─┴─SMTP──server_name──┬─────────────────┬─┬────────────┬─┴───┤
   │                      └─CLASS=──class───┘ └─DEST=──node┘     │
   ├─TRANSLATETABLE──name────────────────────────────────────────┤
   └─XLATETABLE──name────────────────────────────────────────────┘
```

local_options:

```
   ┌─CLASS=A or B──┐
├──┴─CLASS=──class─┴──┬────────────────┬──────────────────────────┤
                      └─OUTPUT=name────┘
```

nje_options:

```
   ┌─CLASS=A or B──┐  ┌─DEST=──local_node─┐  ┌─IDENTIFIER=SYSTEM──────┐
├──┴─CLASS=──class─┴──┴─DEST=──node───────┴──┴─IDENTIFIER=──user_id─┴──►

   ┌─PRIORITY=50─────────┐
►──┴─PRIORITY=──priority─┴──┬───────────────┬─────────────────────────┤
                            └─OUTPUT=name───┘
```

Syntax

```
►►──NLSTRANSLATE──┬─BIG5──┬─────────────────────────┬──────┬──────────►◄
                  │       └─(─┬──────────────────┬──┘      │
                  │           │       ┌─ASCII──┐ │         │
                  │           └─SOSI──┼────────┼─┘         │
                  │                   ├─EBCDIC─┤           │
                  │                   └─SPACE──┘           │
                  ├─EUckanji──┬─────────────────────────┬──┤
                  │           └─(─┬──────────────────┬──┘  │
                  │               │       ┌─ASCII──┐ │     │
                  │               └─SOSI──┼────────┼─┘     │
                  │                       ├─EBCDIC─┤       │
                  │                       └─SPACE──┘       │
                  ├─HAngeul──┬─────────────────────────┬───┤
                  │          └─(─┬──────────────────┬──┘   │
                  │              │       ┌─ASCII──┐ │      │
                  │              └─SOSI──┼────────┼─┘      │
                  │                      ├─EBCDIC─┤        │
                  │                      └─SPACE──┘        │
                  ├─Ibmkanji───────────────────────────────┤
                  ├─JIS78kj──┬─────────────────────┬───────┤
                  │          └─(─┬──────────────┬──┘       │
                  │              │ ┌─ASCII────┐ │          │
                  │              └─┼──────────┼─┘          │
                  │                └─JISROMAN─┘            │
                  ├─JIS83kj──┬─────────────────────┬───────┤
                  │          └─(─┬──────────────┬──┘       │
                  │              │ ┌─ASCII────┐ │          │
                  │              └─┼──────────┼─┘          │
                  │                └─JISROMAN─┘            │
                  ├─Ksc5601──┬─────────────────────────┬───┤
                  │          └─(─┬──────────────────┬──┘   │
                  │              │       ┌─ASCII──┐ │      │
                  │              └─SOSI──┼────────┼─┘      │
                  │                      ├─EBCDIC─┤        │
                  │                      └─SPACE──┘        │
                  ├─SChinese──┬─────────────────────────┬──┤
                  │           └─(─┬──────────────────┬──┘  │
                  │               │       ┌─ASCII──┐ │     │
                  │               └─SOSI──┼────────┼─┘     │
                  │                       ├─EBCDIC─┤       │
                  │                       └─SPACE──┘       │
                  ├─SJiskanji──┬─────────────────────────┬─┤
                  │            └─(─┬──────────────────┬──┘ │
                  │                │       ┌─ASCII──┐ │    │
                  │                └─SOSI──┼────────┼─┘    │
                  │                        ├─EBCDIC─┤      │
                  │                        └─SPACE──┘      │
                  └─TChinese──┬─────────────────────────┬──┘
                              └─(─┬──────────────────┬──┘
                                  │       ┌─ASCII──┐ │
                                  └─SOSI──┼────────┼─┘
                                          ├─EBCDIC─┤
                                          └─SPACE──┘
```

Parameters

name

The service name must be one to eight characters in length. This value is case sensitive.

Restriction: Only characters permitted in MVS data set names are valid.

PRINTER
Specifies the service is to a printer. For LOCAL or NJE devices, the JES spool file created is allocated with RECFM=UM and machine carriage control is written in column 1 of the file. For filter 1, this is always a single space ('09'X). For other filters, it is determined from the data received.

RECFMU
Specifies the service is to a printer. For LOCAL or NJE devices, the JES spool file created is allocated with RECFM=U and carriage control characters are not added to the beginning of each line. The JES spool file is like PRINTER, but carriage control is not added.

RECFMUA
Specifies the service is to a printer. For LOCAL or NJE devices, the JES spool file created is allocated with RECFM=UA. The carriage control (CC) character is taken from the first column of user data after any LPD processing.

Restriction: Only filter 1 should be allowed with this device type. Specify filters 1 in the SERVICE statement so LPD does not print jobs requesting other filter options. The LPD trace would show message EZA0801I for these aborted jobs.

PUNCH
Specifies the service is to a punch device. For LOCAL or NJE devices, the JES spool file created is allocated with RECFM=UM and machine carriage control is written in column 1 of the file.

NONE
Specifies that the service is not currently in use.

LOCAL
Specifies that the data sets sent to a service are written to the local MVS printer or punch.

CLASS=*class*
The SYSOUT class. The default is A for printers and B for punches.

OUTPUT=*name*
Specifies the name of an OUTPUT DD statement that contains additional spool parameters.

NJE
Specifies that the data sets sent to a service are delivered to the NJE system.

CLASS=*class*
The SYSOUT class. The default is A for printers and B for punches.

DEST=*node*
The name of the NJE node. The default is the local node.

IDENTIFIER=*user_id*
The device user ID. The default is SYSTEM.

PRIORITY=*priority*
Specifies the transmission priority. The default is 50.

OUTPUT=*name*
Specifies the name of an OUTPUT DD statement that contains additional spool parameters.

REMOTE
Specifies that data sets (jobs) sent to this service queue are forwarded immediately to the specified remote printer. If the remote printer is not available, the job is discarded.

Guideline: If discarded jobs are a problem, consider sending the jobs directly to the final LPD with LPR, instead of using the MVS LPD as an intermediate router.

printer@host
The destination printer at a specified Internet host. This can be an Internet name or an IP address.

EPORT
For the *Remote* service, when LPR ports 721-731 are in use, LPD tries to use non-reserved ports in the 732 - 1 023 range. The default action, when EPORT is not specified, is to only use the ports 721 - 731 defined by RFC 1179.

EXIT
Specifies any program to be executed before closing, but after allocating and opening, an output data set.

START
Specifies that the program is invoked after allocating and opening the output data set, but before anything is written to the data set. This parameter is mutually exclusive of the END parameter.

END
Specifies that the program is invoked just before closing the output data set. This parameter is mutually exclusive of the START parameter.

program
Name of the program to be invoked. See z/OS Communications Server: IP Configuration Guide for information about using the default LPBANNER or creating your own banner program. The library containing the program should be in the system's link list (LNKLSTxx), or a STEPLIB definition can be used if the library is APF authorized.

FAILEDJOB
Specifies whether a notice of failed jobs should be mailed to users or a job is discarded without notice.

MAIL
Specifies that notices of failed jobs are mailed to users.

Requirement: To use the MAIL parameter, you must also specify the SMTP parameter. Messages are logged in the LPD joblog, showing the information sent to SMTP.

DISCARD
Specifies that failed jobs are discarded without notice.

FILTERS
The control file received by LPD specifies the filter actually used. LPD formatting for each possible filter is described here. When lASf is specified, any filter l received is treated as filter f, described as follows:

f Print formatted file paginates the data set at the size of the page given. It also truncates lines if they exceed a maximum length.

This filter causes the data file to be printed as a plain text file, providing page breaks as necessary.

Restriction: Only the following ASCII control characters are honored:

- HT
- CR
- FF
- LF
- VT
- BS

They are removed from the data stream (not printed) and changed into equivalent spacing and machine carriage control. Any ASCII code that translates to an EBCDIC NL is also honored. However, standard ASCII tables do not have an NL (new line) control character.

JES writers start each job on a new page. Therefore, LPD suppresses any FF (form feed) at the beginning of the data to avoid an extra page eject before the user's data set is printed.

l Print file leaving control characters does not insert pagination but does truncate lines. All lines are single spaced.

This filter causes the specified data file to be printed without filtering out control characters (except LF, which is sed to determine line endings when converting to a JES record oriented spool file). Other ASCII control characters are translated to EBCDIC and printed as text. They are*not* converted to equivalent machine carriage control. Use filter f to ontrol codes like FF and HT to be honored.

Filter l can behave like filter lASf if you specify lASf instead of l. See lASf below.

lASf

p **- Print file with 'pr' format**
Paginates data set, adding titles, the date, and page numbers as well as providing line truncation.

This filter causes the data file to be printed with a heading, page numbers, and pagination. Page breaks are determined by the PAGESIZE configuration on the SERVICE statement, or by ASCII FF (form feed) control characters in the data stream. PAGESIZE includes the title lines printed.

JES writers start each job on a new page. Therefore, LPD suppresses any FF (form feed) at the beginning of the data to avoid an extra page eject before the user's data set is printed.

r **- File to print with FORTRAN carriage control**
Prints the data set, interpreting the first column of each line as a FORTRAN carriage control. The FORTRAN controls are removed from the data stream and translated into equivalent machine carriage control. LPD honors " ", "1", "0", "+", and "-". Other values in column 1 cause single spacing. LPD also truncates lines if they exceed LINESIZE. Page breaks are determined by the PAGESIZE configuration as well as the Fortran controls in column 1.

LINESIZE
Specifies the line length used by the filters when they truncate lines. This statement only applies to services that are designated as either LOCAL or NJE on PAGESIZE (for example, 100 000).

length
The number of characters in a line on a page. Lines longer than this number are truncated. The default is 132.

PAGESIZE
Specifies the page length used by the filters when they paginate.

This statement only applies to services that are designated as either LOCAL or NJE.

lines
The number of lines on a page. The default is 60.

RACF
Controls which users print data sets on this service.

SMTP
Specifies the SMTP server name, CLASS, and DEST options for sending failed jobs notices. For additional information, see the description of the FAILEDJOB MAIL parameter.

server_name
Specifies the name of the SMTP server. If this statement is omitted, the default is SMTP.

CLASS=*class*
The SYSOUT class. The default is A for printers and B for punches.

DEST=*node*
The NJE node to which SMTP messages should be sent.

TRANSLATETABLE
Specifies the translation table in the *name*.TCPXLBIN data set to be used by the client. XLATETABLE is a synonym for this parameter.

name
Specifies the SBCS translate table to be used when a client selects this SERVICE. The name parameter is preceded by either the job name or the *hlq* and followed by TCPXLBIN to form the data set name of the translate table (jobname.name.TCPXLBIN or *hlq*.name.TCPXLBIN). If both data sets exist, which one to use is determined by a search order hierarchy.

See z/OS Communications Server: IP Configuration Guide for more information about search order hierarchy, loading, and customizing of SBCS translation tables.

Tip: XLatetable is a synonym for this option.

XLATETABLE
Specifies the translation table in the *name*.TCPXLBIN data set to be used by the client. TRANSLATETABLE is a synonym for this parameter.

name
Specifies the SBCS translate table to be used when a client selects this SERVICE. The name parameter is preceded by either the job name or the *hlq* and followed by TCPXLBIN to form the data set name of the translate table (jobname.name.TCPXLBIN or *hlq*.name.TCPXLBIN). If both data sets exist, which one to used is determined by a search order hierarchy.

See z/OS Communications Server: IP Configuration Guide for more information about search order hierarchy, loading, and customizing of SBCS translation tables.

Tip: TRANslatetable is a synonym for this option.

NLSTRANSLATE
Specifies the DBCS translation type to be used when a client selects the named SERVICE.

BIG5
Select the translation type from Big-5 P-C DBCS codes to Traditional Chinese host codes.

EUckanji
Select the translation type from Japanese EUC DBCS codes to Japanese host codes.

HAngeul
Select the translation type from Korean PC DBCS codes to Korean host codes.

Ibmkanji
This option causes no conversion to be performed; in other words, data is sent to a printer without translation. Ibmkanji can be used for sending data in EBCDIC. If you select this option, be sure other printers on the same network are all configured with Ibmkanji.

JIS78kj
Select the translation type from JIS 1978 DBCS codes to Japanese host codes. The Escape Sequence, `ESC 2/4 4/0`, is used to express JIS X0208 1978.

JIS83kj
Select the translation type from JIS 1983 DBCS codes to Japanese host codes. The Escape Sequence, `ESC 2/4 4/2`, is used to express JIS X0208 1983.

Ksc5601
Select the translation type from IBM KS DBCS codes to Korean host codes.

SChinese
Select the translation type from Simplified PC Chinese DBCS codes to Simplified Chinese host codes.

SJiskanji
Select the translation type from Shift JIS DBCS codes to Japanese host codes.

TChinese
Select the translation type from Traditional Chinese 5550 PC DBCS codes to Traditional Chinese host codes.

SOSI
Shift-Out and Shift-In characters X'1E' and X'1F' are used in data to delimit DBCS strings.

SOSI ASCII
Shift-Out and Shift-In characters X'1E' and X'1F' are used in data to delimit DBCS strings.

SOSI EBCDIC
Shift-Out and Shift-In characters X'0E' and X'0F' are used in data to delimit DBCS strings.

SOSI SPACE
Shift-Out and Shift-In characters X'20' and X'20' are used in data to delimit DBCS strings.

ASCII (with JIS78KJ and JIS83KJ only)
The ASCII Escape Sequence, `ESC 2/8 4/2`, is used in data to express SBCS strings.

JISROMAN (with JIS78KJ and JIS83KJ only)
The JISROMAN Escape Sequence, `ESC 2/8 4/10`, is used in data to express SBCS strings.

Examples

- PRINTER and PUNCH definitions

  The sample configuration data set SEZAINST(LPDDATA) provides examples of SERVICE statements for LOCAL, REMOTE, and NJE printers and a LOCAL punch.
- EXIT Parameter

  To make the LPBANNER program print a page at the beginning of the printed output, use the EXIT START parameter within a SERVICE statement, as shown here:

  ```
  SERVICE locprt1 PRINTER
      LOCAL
      FILTERS f l p r
      LINESIZE 132
      PAGESIZE 60
      EXIT START LPBANNER
  ```

  To make the LPBANNER program print a page at the end of the printed output, use the EXIT END parameter within a SERVICE statement, as shown here:

  ```
  SERVICE locprt1 PRINTER
      LOCAL
      FILTERS f l p r
      LINESIZE 132
      PAGESIZE 60
      EXIT END LPBANNER
  ```

  See RFC 1179, Section 7.5 Line Printer Daemon Protocol, for more information about the LPD user exit.

Usage notes

- For remote printers, observe these guidelines:

  Remote printers do not require specifications for EXIT, FAILEDJOB, FILTERS, LINESIZE, PAGESIZE, RACF, SMTP, and translation tables. These are defined on the remote system.

  The LPR command must be specified with the printer name as it is specified on the SERVICE statement. The HOST parameter can HOSTNAME or the IP address of the host the LPD is running on, *not* the printer name and IP address of the remote printer.

  ```
  LPR fn (p pebprt h LPDSrvHostName
  ```

  This is required if you want to send data to the remote printer using this LPD.
- With RACF, observe these guidelines:

  In order to print data sets on a printer that has RACF specified, the user must use the JOB option with a valid password on the LPR command.

  If the RACF keyword is specified for the service and a valid password is not supplied, the job sent to that service fails.

If a printer is defined as RACF for a local service on one system and as an NJE service on other systems, then you must specify the RACF keyword on the SERVICE statement on each of the systems where this service is defined.

- For SMTP, observe these guidelines:
  - SMTP is used in conjunction with the FAILEDJOB statement. If the MAIL keyword is used on the FAILEDJOB statement, then the SMTP *server_name* should be set to the name of the SMTP server and an optional CLASS and Destination NJE node.
  - When an attempted print job fails and the MAIL keyword is used on the FAILEDJOB statement, then the LPD server sends a notice of the failure to the SMTP server. This notice is then forwarded to the user ID that submitted the print request.
- For FAILEDJOB, observe these guidelines:

  If the MAIL parameter is specified for any service, then the SMTP statement must be included in the LPD configuration data set.
- For EXIT, observe these guidelines:
  - If the job name is not specified on the corresponding LPR operation, JOB is the data set name that was printed by LPD.
  - If CLASS is omitted on the LPR operation, it contains the sending system's host name.
  - The following parameters are passed to the *program* but not defined in the EXIT statement.

    param1
    : A pointer to a full word return code.

    param2
    : A Pascal string containing the DD name of the spool file, the data set name of the control file, the printer name, and the total number of bytes in the print job. The first two bytes of the Pascal string are the number of bytes of character data starting at byte 3.

    param3
    : A pointer to an open DCB for the JES spool file. The DCB is (DSORG=PS, MACRF=PL, RECFM=UM) for SERVICE printer or SERVICE PUNCH devices. THE DCB is (DSORG=PS, MACRF=PL, RECFM=U) for SERVICE RECFMU devices.

STEPLIMIT statement

Use the STEPLIMIT statement to limit the number of data files and configuration files allowed in a job received by LPD. Jobs that are too complex are rejected with a NACK and are not printed.

Syntax

```
              ┌──────────┐
              ▼          │
►►─STEPLIMIT──────limit──┴──────────────────────────────────────►◄
```

Parameters

limit
: An integer specifying the maximum number of data files and configuration

files allowed in a single job received by LPD. When a wildcard is used in the filename with LPR in some systems, the files are combined into one complex job with many data files.

Usage notes

- Each data file and control file requires a temporary data set on MVS. Each requires memory for control blocks and I/O buffers.
- STEPLIMIT defaults to the preferred value 80 when the keyword is not specified. Increasing this value might cause memory allocation problems with certain system configurations.
- If LPD runs out of memory, reduce the value of either JOBPACING or STEPLIMIT.

UNIT statement

Use the UNIT statement to specify the specific type of DASD where LPD writes its temporary data sets while the transfer of data from an LPR client occurs.

Syntax

```
                ┌──────────────┐
                ▼              │
►►──UNIT────────dasdname───────┴──────────────────────────────────────►◄
```

Parameters

dasdname
The generic name of a group of DASD.

VOLUME statement

Use the VOLUME statement to specify the specific DASD volume where LPD writes its temporary data sets while the transfer of data from an LPR client occurs.

Syntax

```
►►──VOLUME──dasdname──────────────────────────────────────────────────►◄
```

Parameters

dasdname
The volume serial number. The value specified for *name* is case sensitive.

Examples

To set the volume name for new data set to WRKLB4, code the following:

```
VOLUME WRKLB4
```

Chapter 26. PORTMAP and UNIX PORTMAP

This topic contains the following information:
- "PORTMAP cataloged procedure (PORTPROC)"
- "UNIX PORTMAP cataloged procedure (OPORTRPC)"

PORTMAP cataloged procedure (PORTPROC)

This following sample shows the PORTMAP cataloged procedure (PORTPROC).

```
//PORTMAP  PROC MODULE=PORTMAP,PARMS=''
//*
//*        z/OS Communications Server
//*        SMP/E Distribution Name: SEZAINST(PORTPROC)
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5647-A01
//*               (C) Copyright IBM Corp. 1989, 2001
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R2
//*
//PMAP     EXEC PGM=&MODULE,
//           PARM='&PARMS',REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here.  If you add
//*        them to STEPLIB, they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//*        The SYSMDUMP DD statement will cause MVS to provide
//*        an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA
//*        when no GLOBALTCPIPDATA statement is configured.
//*        See the IP Configuration Guide for information on
//*        the TCPIP.DATA search order.
//*        The data set can be any sequential data set or a member of
//*        a partitioned data set (PDS).
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Figure 49. PORTMAP cataloged procedure (PORTPROC)

UNIX PORTMAP cataloged procedure (OPORTRPC)

This following sample shows the UNIX PORTMAP cataloged procedure (OPORTRPC).

```
//PORTMAP  PROC
//*
//*   TCP/IP for MVS
//*   SMP/E distribution name: EZBOPORT
//*
//*   5694-A01  (C) Copyright IBM Corp. 1997, 2001.
//*   Licensed Materials - Property of IBM
//*   This product contains "Restricted Materials of IBM"
//*   All rights reserved.
//*   US Government Users Restricted Rights -
//*   Use, duplication or disclosure restricted by
//*   GSA ADP Schedule Contract with IBM Corp.
//*   See IBM Copyright Instructions.
//*
//*   Function: Unix System Services MVS Portmapper Server main process
//*
//PORTMAP  EXEC PGM=OPORTMAP,REGION=4096K,TIME=1440,
//           PARM='POSIX(ON),ALL31(ON)/'
//*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//         PEND
```

Figure 50. UNIX PORTMAP cataloged procedure (OPORTRPC)

Chapter 27. RPCBIND

This topic includes the following information:
- "RPCBIND cataloged procedure"

RPCBIND cataloged procedure

The following sample shows the RPCBIND cataloged procedure.

```
//RPCBIND  PROC
//*
//*   TCP/IP FOR MVS
//*   SMP/E DISTRIBUTION NAME: EZARBBND
//*
//*   5694-A01  (C) COPYRIGHT IBM CORP. 2007
//*   LICENSED MATERIALS - PROPERTY OF IBM
//*   THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM"
//*   ALL RIGHTS RESERVED.
//*   US GOVERNMENT USERS RESTRICTED RIGHTS -
//*   USE, DUPLICATION OR DISCLOSURE RESTRICTED BY
//*   GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
//*   SEE IBM COPYRIGHT INSTRUCTIONS.
//*
//*   FUNCTION: UNIX SYSTEM SERVICES RPCBIND SERVER MAIN PROCESS
//*
//RPCBIND  EXEC PGM=RPCBIND,REGION=4096K,TIME=1440,
//         PARM=('ENVAR("TZ=EST5EDT")/-dl')
//*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//         PEND
```

Figure 51. Sample RPCBIND

You can specify the following options when starting rpcbind:
- For debug options, you can specify the following -d options to cause rpcbind to send trace information to the daemon facility of syslogd:

  -df Sends non-XDR flow information to syslogd.

  -dl Sends log information of all RPC procedures called to syslogd.

  -dx Sends XDR information to syslogd.
- The -i option enables you to specify the directory where the pid file should be written:

  Rule: The pid filename is always rpcbind.pid. If -i is not specified, the rpcbind process ID is written to /etc/rpcbind.pid.
- The -n option enables you to direct rpcbind to run in a nonswappable environment. A process might need to run non-swappable to ensure it is available during periods of high CPU usage. However, a nonswappable process might convert real storage in the system to preferred storage. Because preferred storage cannot be configured offline, running rpcbind in a non-swappable state can reduce your installation's ability to reconfigure storage in the future.

If you do specify the -n option, ensure that the user ID associated with rpcbind has at least READ access to the resource BPX.STOR.SWAP in the FACILITY class.

The default is to start rpcbind as swappable.

- The -s option specifies the number of statistics entries per binding protocol that rpcbind maintains. The allowable range is 113 - 500. Statistics maintained by the rpcbind server are used to reply to the RPCBPROC_GETSTAT request. See RFC 1833 for more information about statistics maintained by the rpcbind server.

  Result: Rpcbind calculates the number of pages needed to store statistics for the value specified and obtains that number of pages of shared memory for statistics. Thus, rpcbind rounds up the number of statistics entries it tracks to fully use the shared memory.

  Tip: Rpcbind does not start unless it can obtain sufficient shared memory to maintain statistics for the number of entries specified. Configure the number of pages of shared memory available to z/OS with the IPCSHMMPAGES parameter in the BPXPRMxx member of SYS1.PARMLIB.
- To display help information, specify the -? option.

Chapter 28. NCS Interface

This topic contains the following information:
- "NRGLBD cataloged procedure (NRGLBD)"
- "LLBD cataloged procedure (LLBD)" on page 1314

NRGLBD cataloged procedure (NRGLBD)

Update the NRGLBD cataloged procedure by copying the sample provided in SEZAINST(NRGLBD) to your system or recognized PROCLIB and modifying it to suit your local conditions.

Following is the sample NRGLBD cataloged procedure:

```
//NRGLBD   PROC MODULE=NRGLBD,PARMS=''
//*
//*
//*        z/OS Communications Server
//*        SMP/E Distribution Name: EZAEB02D
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5647-A01
//*               (C) Copyright IBM Corp. 1992, 2001
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R2
//*
//NRGLBD   EXEC PGM=&MODULE,
//         PARM='&PARMS',REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here.  If you add
//*        them to STEPLIB, they must be APF authorized. Change
//*        the name as appropriate for your installation.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//*        The SYSMDUMP DD statement will cause MVS to provide
//*        an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA
//*        when no GLOBALTCPIPDATA statement is configured.
//*        See the IP Configuration Guide for information on
//*        the TCPIP.DATA search order.
//*        The data set can be any sequential data set or a member of
//*        a partitioned data set (PDS).
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Figure 52. NRGLBD cataloged procedure

LLBD cataloged procedure (LLBD)

Update the LLBD cataloged procedure by copying the sample provided in SEZAINST(LLBD) to your system or recognized PROCLIB and modifying it to suit your local conditions.

Following is the sample LLBD cataloged procedure:

```
//LLBD     PROC MODULE=LLBD,PARMS=''
//*
//*        z/OS Communications Server
//*        SMP/E Distribution Name: SEZAINST(LLBD)
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5647-A01
//*               (C) Copyright IBM Corp. 1992, 2001
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R2
//*
//LLBD     EXEC PGM=&MODULE,
//         PARM='&PARMS',REGION=4096K,TIME=1440
//*
//*        The C runtime libraries should be in the system's link list
//*        or add them to the STEPLIB definition here.  If you add
//*        them to STEPLIB, they must be APF authorized.  Change
//*        the name as appropriate for your installation.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//*
//*        The SYSMDUMP DD statement will cause MVS to provide
//*        an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//SYSIN    DD DUMMY
//*
//*        SYSTCPD explicitly identifies which data set is to be
//*        used to obtain the parameters defined by TCPIP.DATA
//*        when no GLOBALTCPIPDATA statement is configured.
//*        See the IP Configuration Guide for information on
//*        the TCPIP.DATA search order.
//*        The data set can be any sequential data set or a member of
//*        a partitioned data set (PDS).
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.SEZAINST(TCPDATA)
```

Figure 53. LLBD cataloged procedure (LLBD)

Chapter 29. SMTP server

This topic contains the following information:

- "SMTP cataloged procedure (SMTPPROC)"
- "Summary of SMTP configuration statements" on page 1316
- "Steps for using the SMTP server exits" on page 1319
- "SMTP configuration data set statements" on page 1325

SMTP cataloged procedure (SMTPPROC)

Restriction: SMTP does not support z/OS UNIX files.

This procedure contains the data set name for the SMTP configuration data set.

```
//SMTP PROC MODULE=SMTP,DEBUG=,PARMS='NOSPIE/',SYSERR=SYSERR
//*
//* z/OS Communications Server
//* SMP/E Distribution Name: EZAEB017
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5694-A01
//*               Copyright IBM Corp. 1989, 2008
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*
//* Status:       CSV1R10
//*
//SMTP     EXEC PGM=MVPMAIN,
//             PARM='&MODULE,PARM=&DEBUG,ERRFILE(&SYSERR),&PARMS',
//             REGION=6144K,TIME=1440
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*
//*        The SYSMDUMP DD statement will cause MVS to provide
//*        an IPCS readable dump for ABENDs.
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*        SYSPRINT points to a data set used for the output from
//*        internal calls to IDCAMS.  It can be a temporary data set.
//*
//SYSPRINT DD SYSOUT=*
//*
//*        SYSERR contains runtime diagnostics from Pascal.  It can be
//*        a data set or SYSOUT.
//*
//SYSERR   DD SYSOUT=*
//*
//*        SYSDEBUG receives output that is generated when the DEBUG
//*        configuration statement is specified in SMTP.  It can be
//*        a data set or SYSOUT.
//*
//SYSDEBUG DD SYSOUT=*
//*
//*        OUTPUT contains the startup and shutdown messages from SMTP.
//*        It can be a data set or SYSOUT.
//*
//OUTPUT   DD SYSOUT=*
//*
//*        LOGFILE receives output that is generated when the LOG
//*       configuration statement is specified in SMTP.  It can be
//*       a data set or SYSOUT.
//*
//LOGFILE  DD SYSOUT=*
//*
//*       SMTPNJE is the output of the SMTPNJE command.
//*       Before running SMTP you should use the SMTPNJE command
//*       to create the data set and then you can remove the
//*       "*" from the following line to activate SMTPNJE.
//*
//*SMTPNJE  DD DSN=TCPIP.SMTPNJE.HOSTINFO,DISP=SHR
//*
//*       CONFIG points to a sample configuration data set.
//*       Before running SMTP you should modify this file to
//*       include parameters suitable for your installation.
//*
//CONFIG   DD DSN=TCPIP.SEZAINST(SMTPCONF),DISP=SHR
//*
//*       SECTABLE points to your SMTP security table data set.
//*       If you are running with the SECURE option, this data set
//*       will contain a list of NJE users who are authorized to
//*       use the gateway.
//*       You must remove the "*" from the following line to allow
//*       SMTP to find the data set.
//*
//*SECTABLE DD DSN=SMTP.SMTP.SECTABLE,DISP=SHR
//*
//*       SMTPRULE points to the data set containing the rewrite rules
//*       for the header addresses. You must specify REWRITE822HEADER
//*       YES for this data set to be read.
//*       You must remove the "*" from the following line to allow
//*       SMTP to find the data set.
//*
//*SMTPRULE DD DSN=SMTP.SMTP.RULE,DISP=SHR
//*
//*       SYSTCPD explicitly identifies which data set is to be
//*       used to obtain the parameters defined by TCPIP.DATA
//*       when no GLOBALTCPIPDATA statement is configured.
//*       See the IP Configuration Guide for information on
//*       the TCPIP.DATA search order.
//*       The data set can be any sequential data set or a member of
//*       a partitioned data set (PDS).
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Figure 54. SMTP cataloged procedure (SMTPPROC)

Summary of SMTP configuration statements

The SMTP configuration statements are summarized in Table 97.

Table 97. Summary of SMTP configuration statements

Statement	Description	See
ALTNJEDOMAIN	Specifies an alternative domain name of the NJE network, if SMTP is running as a mail gateway.	"ALTNJEDOMAIN statement" on page 1325
ALTTCPHOSTNAME	Specifies an additional host name for the local host. Mail received for this host name is accepted and delivered locally.	"ALTTCPHOSTNAME statement" on page 1326

Table 97. Summary of SMTP configuration statements (continued)

Statement	Description	See
ATSIGN	Specifies that you want SMTP to use this character in the addressing strings generated by SMTP.	“ATSIGN statement” on page 1326
BADSPOOLFILEID	Specifies the user ID on the local system where SMTP transfers unreadable spool files and looping mail.	“BADSPOOLFILEID statement” on page 1327
CHECKSPOOLSIZE	Enables SMTP to check the size of the JES spool file prior to writing the data to the *hlq*.TEMP.NOTE file.	“CHECKSPOOLSIZE statement” on page 1327
DBCS	Specifies that DBCS code conversion be performed on the mail.	“DBCS statement” on page 1328
DEBUG	Records all SMTP commands and replies.	“DEBUG statement” on page 1330
DELETEBADSPOOLFILE	Permits SMTP to delete the spool file from the JES spool that would cause an ABENDS001 when accessed by SMTP.	“DELETEBADSPOOLFILE statement” on page 1330
DISALLOWCMD	Enables SMTP server to discontinue support for certain SMTP commands specified by the customer.	“DISALLOWCMD statement” on page 1331
EXITDIRECTION	Enables SMTP to call the SMTP exit provided by the customer for data coming from the JES spool.	“EXITDIRECTION statement” on page 1332
FINISHOPEN	Specifies the SMTP wait time for connection.	“FINISHOPEN statement” on page 1333
GATEWAY	Specifies operation of SMTP as a gateway.	“GATEWAY statement” on page 1333
INACTIVE	Specifies the SMTP wait time before closing an inactive connection.	“INACTIVE statement” on page 1335
INBOUNDOPENLIMIT	Specifies the maximum number of simultaneous TCP connections over which SMTP server receives mail.	“INBOUNDOPENLIMIT statement” on page 1335
IPMAILERADDRESS	Specifies the IP address of an SMTP server that can resolve network addresses of unknown hosts.	“IPMAILERADDRESS statement” on page 1336
IPMAILERNAME	Enables SMTP to forward non-local mail to the specified IP mailer name.	“IPMAILERNAME statement” on page 1337
LISTENONADDRESS	Allows you to restrict which IP address is used to receive and send mail on a multihomed system.	“LISTENONADDRESS statement” on page 1338
LOCALCLASS	Specifies the spool data set class for local mail delivery.	“LOCALCLASS statement” on page 1339
LOCALFORMAT	Specifies the spool data set format for local host mail delivery.	“LOCALFORMAT statement” on page 1339
LOG	Directs SMTP to log all SMTP traffic.	“LOG statement” on page 1340
MAILER	Specifies the address of the batch SMTP server that receives mail.	“MAILER statement” on page 1340
MAILFILEDSPREFIX	Specifies the prefix to add to mail data sets.	“MAILFILEDSPREFIX statement” on page 1342

Table 97. Summary of SMTP configuration statements (continued)

Statement	Description	See
MAILFILESUNIT	Specifies the unit where SMTP mail data sets reside.	“MAILFILEUNIT statement” on page 1343
MAILFILEVOLUME	Specifies the volume where newly allocated SMTP data sets reside.	“MAILFILEVOLUME statement” on page 1343
MAXMAILBYTES	Specifies the maximum size of mail that is accepted over a TCP connection.	“MAXMAILBYTES statement” on page 1344
MAXMSGSENT	Controls the behavior of the SMTP client code by limiting the number of messages sent on a single TCP/IP connection	“MAXMSGSENT statement” on page 1345
NJECLASS	Specifies the spool data set class for mail delivered on an NJE network.	“NJECLASS statement” on page 1346
NJEDOMAIN	Specifies the domain name of the NJE network if SMTP functions as a gateway.	“NJEDOMAIN statement” on page 1346
NJEFORMAT	Specifies the spool data set format for mail delivered on the NJE network.	“NJEFORMAT statement” on page 1347
NJENODENAME	Specifies the node name of the local JES2 or JES3 node for mail delivered on the NJE network.	“NJENODENAME statement” on page 1348
NOLOG	Turns off the logging of mail transactions.	“NOLOG statement” on page 1349
NOSOURCEROUTE	Controls whether this SMTP generates and passes a source routing string for the originator address (MAILCMD) or for the recipient address (RCPTCMD).	“NOSOURCEROUTE statement” on page 1349
OUTBOUNDOPENLIMIT	Specifies a limit on the maximum number of simultaneous TCP connections over which SMTP actively delivers mail.	“OUTBOUNDOPENLIMIT statement” on page 1351
PORT	Specifies an alternative port number for the SMTP server during testing.	“PORT statement” on page 1351
POSTMASTER	Specifies the address (or addresses) for mail addressed to the postmaster at the local host.	“POSTMASTER statement” on page 1352
RCPTREPLY452	Enables SMTP to handle reply code 452 differently for the RCPT command.	“RCPTREPLY452 statement” on page 1353
RCPTRESPONSEDELAY	Specifies how long the SMTP server delays responding to the RCPT commands.	“RCPTRESPONSEDELAY statement” on page 1353
REMOTEPORT	Specifies which remote port number the SMTP client uses for sending outbound mail.	“REMOTEPORT statement” on page 1354
RESOLVERRETRYINT	Specifies the number of minutes SMTP waits between attempts to resolve domain names.	“RESOLVERRETRYINT statement” on page 1355
RESOLVERUSAGE	Indicates whether SMTP should attempt to resolve non-local domain names.	“RESOLVERUSAGE statement” on page 1355
RESTRICT	Specifies addresses of users who are not allowed to use SMTP mail services.	“RESTRICT statement” on page 1356
RETRYAGE	Specifies the number of days after which mail is returned as undeliverable.	“RETRYAGE statement” on page 1357

Table 97. Summary of SMTP configuration statements (continued)

Statement	Description	See
RETRYINT	Specifies the number of minutes between attempts to send mail to an inactive TCP host.	"RETRYINT statement" on page 1358
REWRITE822HEADER	Prevents SMTP from rewriting RFC 822 headers with source routing.	"REWRITE822HEADER statement" on page 1358
SECURE	Specifies that SMTP operates as a secure mail gateway between TCP network sites and NJE network sites.	"SECURE statement" on page 1359
SMSGAUTHLIST	Specifies the addresses of users authorized to issue privileged SMTP SMSG commands.	"SMSGAUTHLIST statement" on page 1360
SPOOLPOLLINTERVAL	Specifies the interval for SMTP to check the spool for incoming batch data sets.	"SPOOLPOLLINTERVAL statement" on page 1361
STOPONRENF	Controls the behavior of the SMTP server so if a RENAME failure occurs on a data set associated with the batch connection (257), the SMTP server stops by normal termination.	"STOPONRENF statement" on page 1361
TEMPERRORRETRIES	Specifies the number of times SMTP tries to redeliver mail to a host with a temporary problem.	"TEMPERRORRETRIES statement" on page 1362
TIMEZONE	Sets the printable name of the local time zone.	"TIMEZONE statement" on page 1363
WARNINGAGE	Specifies the number of days after which a copy of the mail is returned to the sender, indicating that the mail has so far been undeliverable and that SMTP continues to retry delivery for the number of days specified in RETRYAGE.	"WARNINGAGE statement" on page 1364

Steps for using the SMTP server exits

Use the SMTP server exit to check and subsequently accept or reject mail inbound from a TCP/IP network or mail outbound from the JES spool. For example, you can code an exit to check the MAIL FROM: string on outbound mail or to control the influx of unwanted inbound mail (commonly referred to as *spam*).

Tip: The exits described in this topic are also used by the CSSMTP application for checking outbound mail only. See Chapter 30, "Communications Server SMTP application," on page 1365 for more information.

The SMTP server dynamically determines if an SMTP exit program exists. This determination is based upon the SMTP exit program association with the name token EZBTCPIPSMTPEXIT using the MVS SETPROG command. So, the presence of the SMTP exit program allows the SMTP server to call the exit program for inbound TCP/IP connection data. If you determine that the exit program needs to be called to interrogate data coming from the JES spool, follow these steps:

1. Add the EXITDIRECTION statement and the appropriate parameters to the SMTP configuration. Also, stop and restart the SMTP server in order to recognize the new configuration settings.

2. In order to work correctly with the JES connection, add code to the user exit program.

   Rules:

   - The JES connection ID is always 257.
   - The field (EZBPIPV4) representing the remote IP address is always zero for the JES connection.
   - For TCP/IP connections, the field (EZBPTOKP) representing the SAF token information is always zero.

   If SAF token information is requested, the field EZBPTOKP contains the address of the token. However, this field can be zero if the SMTP server was unable to retrieve the SAF token from JES. The exit program needs to be coded to handle this situation. The SAF token length is 80 bytes and the SAF token version is 1. The SAF token provides information about the submitting user ID and the submitter node of the JES data. This data can be compared to the sender information about the MAIL FROM: string. For more information about what is provided in the SAF token, see the RUTKN information in z/OS Security Server RACF Data Areas.

3. Recompile the user exit program with the version 2 copy of the EZBZSMTP DSECT. This picks up the changes in the parameter list.

4. Ensure that the user ID specified on the POSTMASTER statement in the SMTP configuration file is a valid user ID.

   Requirement: This user ID and host must be able to receive mail.

   When mail is rejected by the SMTP exit program for the JES connection ID, it is always returned to the POSTMASTER. The POSTMASTER must determine what happens to the rejected JES data. After the SMTP exit program rejects the JES data, the entire spool file is rejected, which might include multiple notes. Depending on how the JES data is spooled, this might be a large amount of data.

   The POSTMASTER can modify the data and resend it to SMTP, or it can discard the data. The SMTP exit program policies determine whether or not the POSTMASTER receives large quantities of mail that require review. When the SMTP exit program rejects mail from a TCP/IP connection, the remote SMTP client determines what happens to the rejected mail. In this case, the mail becomes undeliverable and might be returned to the originator.

If you run the exit program in both directions, performance might be impacted.

Tip: If you want the exit program to interrogate only inbound TCP/IP connections, do not make any configuration changes.

If the SMTP server receives mail from a TCP/IP network, and then sends it out on a TCP/IP connection (relaying the mail), the SMTP server invokes the exit program only one time on the inbound path.

The exit should be written in Assembler Language. Standard z/OS Assembler entry and exit linkage must be used. See z/OS MVS Programming: Assembler Services Guide for the linkage conventions.

The exit is invoked with the settings shown in Table 98.

Table 98. SMTP user exit settings

Authorization	Problem state
Dispatchable Unit Mode	Task
Cross memory mode	PASN=HASN
Amode	31–bit
ASC mode	Primary address space control (ASC) mode
Interrupt status	Enabled for interrupts
Locks	Unlocked

On entry to the exit, the register contents are:

Register 0
Used as a work register by the system

Register 1
Address of the exit's input parameter list (see Table 99)

Registers 2-12
Unassigned

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
Address of the exit routine

The exits input parameter list contains the information shown in Table 99. An assembler macro is available to provide you with a DSECT describing this area. The name of the macro is EZBZSMTP and the macro resides in SEZACMAC. It enables an optional label but has no operands. It provides symbolic names for the 3 return codes and 18 action codes. The labels are as shown in Table 99.

Table 99. SMTP server exit input parameter list

Label name	Width/Value	Description
Parameter list variables		
EZBZSMTP		DSECT Name
EZBPVERS (See note 1)	1 Fullword	Version number
EZBPACTN (See note 2)	1 Fullword	Action code
EZBPUSER (See note 3)	1 Fullword	Returned Reg15 of initialization call
EZBPCNID (See note 4)	1 Fullword	Connection ID
EZBPTOKP (See note 17)	1 Fullword	Address of SAF (security) token
	2 Fullwords	Reserved for future use
EZBPIPV4 (See note 6)	1 Fullword	IP addr of remote SMTP
EZBPDLEN (See note 7)	1 Fullword	Length of data in buffer
EZBPBUFF (See note 8)	1 Fullword	Buffer address

Table 99. SMTP server exit input parameter list (continued)

Label name	Width/Value	Description
Parameter list variables		
Constants		
EZBRAGN	0	Return code to continue
EZBRACC	4	Return code to accept mail
EZBRREJ	8	Return code to reject mail
Action codes		
EZBAINIT	1	Initialization call (See note 9)
EZBATERM	2	Termination call (See note 10)
EZBADATA	3	SMTP DATA command
EZBAEXPN	4	SMTP EXPN (expand) command
EZBAHELO	5	SMTP HELO (hello) command
EZBAHELP	6	SMTP HELP command
EZBAMAIL	7	SMTP MAIL command
EZBANOOP	8	SMTP NOOP command (See note 11)
EZBAQUEU	9	IBM SMTP QUEU (queue) command
EZBAQUIT	10	SMTP QUIT command (See note 12)
EZBARCPT	11	SMTP RCPT (recipient) command
EZBARSET	12	SMTP RSET (Reset) command (See note 13)
EZBATICK	13	IBM SMTP TICK command
EZBAVERB	14	IBM SMTP VERB command
EZBAVRFY	15	SMTP VRFY (Verify) command
EZBADBUF	16	Data buffer (See note 14)
EZBAEODB	17	End of data buffers (last chance) (See note 15)
EZBACONN	18	End of connection (See note 16)

Notes:

1. A word containing a version number. The value is one when the exit program is called for INBOUND mail only. The value is two when the exit program is called for BOTH (inbound and outbound) mail.
2. A word-aligned word containing an action code describing the buffer contents (if any).
3. A word containing the user supplied token from the initialization call.

4. A word containing a connection identifier number to distinguish between concurrent connections. The connection ID representing the JES spool data is always 257.
5. Two unused words (reserved space).
6. A word containing the IP address of the connecting remote SMTP. It contains 0 if the connection ID is 257 (JES connection ID).
7. A word containing the actual length of data in the buffer. If the buffer length is meaningless for the action code, the length is set to 0.
8. A word containing a 31–bit address that points to the actual buffer. If the buffer length is 0, this parameter should not be used.
9. Buffer is empty, expect return token in R15.
10. Buffer is empty, application shutting down, exit return code is ignored. This call (and all others) might not occur during abnormal termination.
11. Exit return code is ignored whenever this command is detected.
12. Exit return code is ignored.
13. Exit return code is ignored.
14. Data buffer (there is no command associated with this) approximately 1 024 bytes of data or less. The data are in EBCDIC, but might be in an multicultural support mode (non-English).
15. End of data marker (there is no command associated with this and the buffer contents are meaningless). This is the last chance to reject this message.
16. TCP/IP connection terminated or end of file for JES spool data.
17. If SAF information is requested using the EXITDIRECTION statement in the SMTP configuration, the SMTP server sets this field to a 31-bit address that points to the SAF (security) token information. If the SMTP server was unable to retrieve the SAF token or if the EXITDIRECTION statement is not configured, this field contains 0. For TCP/IP connections, this field is always 0.

There are two control invocations of the SMTP user exit. One for initialization, and the other for termination. On return from the initialization call, the contents of register 15 is treated as a 4-byte user token that is returned on all other exit invocations. See Table 99 on page 1321 for more information. The user token is not used by SMTP, but only passed on subsequent calls to allow a reentrant exit to have static data (using getmain or some other method). It is expected that certain data sets might be read during the initialization call and that tables of known spamming Internet addresses might be constructed at this time for later use. The termination call allows report generation or any other clean-up activity that the exit might do prior to the stopping of SMTPPROC under normal termination logic.

There are three supported return code values which the exit program might set. For certain action codes such as initialization (EZBAINT), termination (EZBATERM) and end of connection (EZBACONN) the return code value is ignored. The returned value and expected meanings are as follows:

0 Call user-supplied exit program again.

4 Accept this message or command and do not call again for this message.

8 Reject message or command and do not call again for this message.
- During processing of SMTP commands, the reply code `550 service denied due to user supplied exit` is generated immediately.

- During note data processing (action code = 16), the reply code `550 service denied due to user supplied exit` is generated when the end of data marker (action code = 17) is received.

Return codes that are not valid are converted to a 0, and the exit is called again.

Tip: Certain commands should not be rejected, because they can cause unpredictable results with the partner SMTP application.

Rule: Certain commands, such as NOOP, QUIT, and RSET should always be accepted.

The connection identifier is a unique number during the life of the connection. You can use it to distinguish between multiple concurrent connections that can be present. Each has its own state information in SMTPPROC, and if the exit wants to keep any state information, this field can be used to keep each message's state separate. Connection identifiers normally become available for reuse after a QUIT command or the end of connection (action code 18), or both, occur. They normally first appear with a HELO command.

The buffer contents for action codes 3 through 15 contain the SMTP command.

The buffer contains data that has been translated using the EBCDIC encoding tables configured for the SMTPPROC. Data buffers might not be in English and might contain NLS characters.

Unknown commands are rejected by SMTPPROC and the exit is *not* called. The buffer normally contains the SMTP command. See RFC 821 for exact spellings and format.

Guideline: The SMTP command can appear in either upper, lower, or mixed case.

The initialization and termination calls do not have a connection number. The return code from the initialization call is not checked, but placed in the ezbpuser field. The return code from the termination call is moot. These calls are always active if the exit is active.

Interaction between SMTP and user exit program

During an active connection, SMTP determines whether the user exit program is called again based on the return code passed back to SMTP from the previous invocation of the exit.

The user exit is not called again for the affected connection until the resetting action codes are received, and only if the ezbracc or the ezbrrej return codes are received from the exit from a connection-oriented call. The accept or reject state might remain in effect for only the current call however.

In Table 100, the following codes are not sent to the exit if the current state is accept or reject and do not change the state.

Table 100. Exit action codes and values (Part 1)

Action code	Value
3	DATA
4	EXPN

Table 100. Exit action codes and values (Part 1) (continued)

Action code	Value
6	HELP
8	NOOP
9	QUEU
11	RCPT
13	TICK
14	VERB
15	VRFY
16	data buffers

In Table 101, the following codes are not sent to the exit, but they ensure that the next command (if any) goes to the exit as it resets the next state to ezbragn.

Table 101. Exit action codes and values (Part 2)

Action code	Value
10	QUIT
12	RSET
17	End of data buffers (final chance)

In Table 102, the following is always sent to the exit if it is active, and the return code received determines the new state.

Table 102. Exit action codes and values (Part 3)

Action code	Value
5	HELO
7	MAIL
18	Connection closed (termination of individual connection). Connection number available for reuse and state is reset to ezbragn.

SMTP configuration data set statements

This topic contains the SMTP configuration data set statements.

ALTNJEDOMAIN statement

Use the ALTNJEDOMAIN statement to specify an alternative domain name of the NJE network when SMTP is running as a mail gateway.

Syntax

```
►►──ALTNJEDOMAIN──domain────────────────────────────────────────────►◄
```

Parameters

domain
: The alternative domain name of the NJE network. The alternative NJE domain name is a 1 - 64 alphanumeric string of characters.

Examples

Using the ALTNJEDOMAIN statement is helpful when the NJE network is known by multiple domain names, such as VNET and VNET.IBM.COM.

```
ALTNJEDOMAIN VNET
```

Usage notes

The ALTNJEDOMAIN statement can be specified only once.

ALTTCPHOSTNAME statement

Use the ALTTCPHOSTNAME statement to specify an alternative, fully-qualified host name by which SMTP recognizes the local host. Mail sent to users at *host_name* are treated as if they were local users. You can use the ALTTCPHOSTNAME statement to specify up to 16 alternative host names.

Syntax

```
►►──ALTTCPHOSTNAME──host_name──────────────────────────────────────────────►◄
```

Parameters

host_name
: The name of the destination host.

Examples

In the following example, mail sent to users at PALACE are treated as if they were local users:

```
ALTTCPHOSTNAME PALACE
```

ATSIGN statement

Use the ATSIGN statement to cause SMTP to use this character in the addressing strings generated by SMTP.

Tip: This might affect operability between sites using different code pages.

Syntax

```
►►──ATSIGN──symbol─────────────────────────────────────────────────────────►◄
```

Parameters

symbol
: The input symbol, which is a single-byte representation of the @ symbol in a national language code page.

Usage notes

- If this statement is not specified, SMTP defaults to the @ symbol (defined as a value '7C'). For details about EBCDIC character set definitions, see *3174 Character Set Reference*.
- The ATSIGN statement cannot be used in combination with the REWRITE822HEADER statement. REWRITE822HEADER defaults to YES, and you must set this to NO in your SMTP configuration file if you want to use ATSIGN. The REWRITE822HEADER statement must be coded before the ATSIGN statement.
- If the ATSIGN statement was used previously, and you are assigning a new symbol for that statement, ensure that all the mail has been sent. If old mail exists that is using the old symbol, then problems might occur.

BADSPOOLFILEID statement

Use the BADSPOOLFILEID statement to specify the user ID on the local system where SMTP transfers unreadable spool files and looping mail.

Syntax

```
      ┌─BADSPOOLFILEID TCPMAINT──┐
►►────┤                          ├──────────────────────────────────────►◄
      └─BADSPOOLFILEID ──user_id─┘
```

Parameters

user_id
: The user ID on the local system where bad spool files and looping mail are delivered. The user ID should be a maximum of eight characters. The default is TCPMAINT. If RACF is active, then a RACF profile must be defined for this user ID.

Examples

In this example, unreadable spool files and looping mail are transferred to the user ID, DBARTON.

```
BADSPOOLFILEID DBARTON
```

Usage notes

The BADSPOOLFILEID statement can be specified only once.

CHECKSPOOLSIZE statement

Use the CHECKSPOOLSIZE statement to specify cause SMTP to check the size of JES spool file. If the JES spool file is larger than the primary allocation for the *hlq*.TEMP.NOTE data set, the resulting SMTP note is truncated. When the SMTP note is truncated, an informational message EZA5340I or EZA5342I is generated in the SMTP OUTPUT file.

The choice of which message is used depends on the format of the JES spool file (NETDATA or batch) being read. The truncated SMTP note also has the corresponding message appended to the bottom of the note. This enables the system administrator to correlate the SMTP note with the SMTP mailer doing the truncation and increase the MAXMAILBYTES if it is appropriate. The SMTP mailer continues to process mail. The default for this parameter is disabled.

Syntax

```
►►──CHECKSPOOLSIZE─────────────────────────────────────────────────────►◄
```

Parameters

There are no parameters for this statement.

Usage notes

If this parameter is not specified, SMTP works as originally designed. Secondary allocations are requested for continuing growth of the *hlq*.TEMP.NOTE data set. Abend B37 can occur if 16 extents are exceeded and more storage is needed.

Related topics

"MAXMAILBYTES statement" on page 1344

DBCS statement

Use the DBCS statement to specify that SMTP should perform DBCS code conversion on the mail. The parameters of the DBCS statement determine which translation table should be used in the conversion.

Syntax

```
►►──DBCS──┬──┬─JIS78KJ─┬──┬─ASCII────┬──┬──────────────────────────────►◄
          │  └─JIS83KJ─┘  └─JISROMAN─┘  │
          ├─BIG5────────────────────────┤
          ├─EUCKANJI────────────────────┤
          ├─IBMKANJI────────────────────┤
          ├─HANGEUL─────────────────────┤
          ├─KSC5601─────────────────────┤
          ├─SCHINESE────────────────────┤
          ├─SJISKANJI───────────────────┤
          └─TCHINESE────────────────────┘
```

Parameters

JIS78KJ
Specify JIS78KJ if the conversion between IBM Kanji and JIS 1978 DBCS codes is to be performed. The Escape Sequence for JIS X0208 1978 is `ESC 2/4 4/0`. SMTP loads the JIS78KJ DBCS translation table from the TCPKJBIN binary translate table data set.

Restriction: When JIS78KJ and JIS83KJ are used, either ASCII or JISROMAN must be used or an error occurs and SMTP ends. SMTP configuration reads the next parameter in the configuration file as the third DBCS statement entry.

JIS83KJ
Specify JIS83KJ if the conversion between IBM Kanji and JIS 1983 DBCS codes

is to be performed. The Escape Sequence for JIS X0208 1983 is `ESC 2/4 4/2`. SMTP loads the JIS83KJ DBCS translation table from the TCPKJBIN binary translate table data set.

Restriction: When JIS78KJ and JIS83KJ are used, either ASCII or JISROMAN must be used or an error occurs and SMTP ends. SMTP configuration reads the next parameter in the configuration file as the third DBCS statement entry.

ASCII
Specify ASCII for JIS78KJ or JIS83KJ if the mail is shifted in ASCII code from JIS Kanji code. The Escape Sequence for ASCII is `ESC 2/8 4/2`.

JISROMAN
Specify JISROMAN for JIS78KJ or JIS83KJ if the mail is shifted in JISRoman code from JIS Kanji code. The Escape Sequence for JISRoman is `ESC 2/8 4/10`.

BIG5
Specify BIG5 if the conversion between IBM Traditional Chinese host DBCS codes and Big-5 PC DBCS codes is to be performed. SMTP loads the BIG5 DBCS translation table from the TCPCHBIN binary translate table data set.

EUCKANJI
Specify EUCKANJI if the conversion between IBM Kanji and Japanese EUC DBCS codes is to be performed. SMTP loads the EUCKANJI DBCS translation table from the TCPKJBIN binary translate table data set.

IBMKANJI
Specify IBMKANJI if IBM (EBCDIC) Kanji conversion is to be used. This option causes *no* conversion to be performed on the body of the mail. This can be used for the sending and receiving of mail in EBCDIC.

Guideline: If this option is selected, other SMTP servers on the same network should all be configured with IBMKANJI. If IBMKANJI is specified, and LOCALFORMAT or RSCSFORMAT is set to PUNCH, then mail received in ASCII can be folded to inconsistent record lengths. In this case, LOCALFORMAT and RSCSFORMAT should be set to NETDATA.

The IBMKANJI transfer type does not require any translate table to be loaded.

HANGEUL
Specify HANGEUL if the conversion between IBM Korean host DBCS codes and Korean PC DBCS codes is to be performed. SMTP loads the HANGEUL DBCS translation table from the TCPHGBIN binary translate table data set.

KSC5601
Specify KSC5601 if the conversion between IBM Korean host DBCS codes and IBM KS DBCS codes is to be performed. SMTP loads the KSC5601 DBCS translation table from the TCPHGBIN binary translate table data set.

SCHINESE
Specify SCHINESE if the conversion between IBM Simplified Chinese host DBCS codes and Simplified Chinese PC DBCS codes is to be performed. SMTP loads the SCHINESE DBCS translation table from the TCPSCBIN binary translate table data set.

SJISKANJI
Specify SJISKANJI if the conversion between IBM Kanji and Shift JIS DBCS codes is to be performed. SMTP loads the SJISKANJI DBCS translation table from the TCPKJBIN binary translate table data set.

TCHINESE
Specify TCHINESE if the conversion between IBM Traditional Chinese host

DBCS codes and Traditional Chinese 5550 PC DBCS codes is to be performed. SMTP loads the TCHINESE (5550) DBCS translation table from the TCPCHBIN binary translate table data set.

Examples

In the following example, IBM Traditional-Chinese-to Traditional-Chinese 5550 PC code conversion is used:

```
DBCS TCHINESE
```

Usage notes

- The transmission of DBCS mail by SMTP uses two different translation tables, SBCS and DBCS. SBCS characters in the mail headers and in the mail body are converted using either *hlq*.STANDARD.TCPKJBIN, TCPHGBIN, TCPSCBIN, or TCPCHBIN.
- DBCS conversion is only performed on outgoing and incoming mail to and from other hosts. Mail spooled to SMTP (for example, using SMTPNOTE) for the local host is delivered directly, without DBCS code conversion.

Related topics

Appendix A, “Translation tables,” on page 1417

DEBUG statement

Use the DEBUG statement to record SMTP commands and replies in the SMTP debug data set (which is pointed to by the SYSDEBUG DD statement).

Syntax

```
►►──DEBUG──────────────────────────────────────────────────────────────►◄
```

Parameters

There are no parameters for this statement.

Usage notes

The SMTP connection number is recorded along with each SMTP command or reply. The connection numbers are used as follows:

- Connection numbers 0 through 255 are used for SMTP connections over a TCP network.
- Connection number 257 is used for the batch SMTP connection.

DELETEBADSPOOLFILE statement

Use the DELETEBADSPOOLFILE statement to change the behavior of SMTP when it detects a spool file on the JES spool that would cause an ABENDS001 on the JES spool. If not coded, the default behavior is for SMTP to generate error message EZA5469E and terminate. Then, the system administrator can examine the offending spool file. The application generating the spool file should be changed. While that is being done, the DELETEBADSPOOLFILE statement can be used so SMTP continues to run and automatically deletes any offending spool file.

Tip: These spool files might contain customer data; therefore, it is the system administrator's responsibility to give SMTP permission to do automatic deletion. If a spool file is automatically deleted, SMTP generates message EZA5470E to alert the user.

Syntax

```
►►──DELETEBADSPOOLFILE──────────────────────────────────────────────►◄
```

Parameters

There are no parameters for this statement.

Examples

Use the following code to cause SMTP to delete any spool file it detects on the JES spool that causes an ABENDS001 when accessed:

```
DELETEBADSPOOLFILE
```

DISALLOWCMD statement

Use the DISALLOWCMD statement to control whether this SMTP server does not support these SMTP commands. Each SMTP command has a decimal number, specifying that number after the DISALLOWCMD statement causes that SMTP command to no longer be supported. The SMTP server responds with the reply code `502 Command <SMTP command>` not implemented. The SMTP server responds this way whether or not the SMTP command is being issued by way of the TCP/IP connection or the JES spool.

Restriction: You can disallow only the following SMTP commands:

- VERB
- QUEU
- HELP
- VRFY
- EXPN

If the number parameter following the DISALLOWCMD statement is not valid for any reason, then the SMTP server default behavior is to allow these SMTP commands and respond appropriately.

Restriction: Code this statement only one time in the SMTP configuration file, because the last instance of this statement is the only one that is used.

Syntax

```
►►──DISALLOWCMD──(number)──────────────────────────────────────────►◄
```

Parameters

number
Indicates what SMTP commands should not be allowed. Multiple SMTP commands can be disallowed by adding the SMTP command number together. The following decimal numbers are assigned to each command:

```
VERB= 1, QUEU= 2, HELP= 4, VRFY= 8 and EXPN= 16
```

A valid decimal number parameter ranges from 1 to 31.

Examples

Use the following code to specify that the SMTP commands, VERB and VRFY, are no longer supported:

```
DISALLOWCMD   9
```

Use the following code to specify that the SMTP commands, HELP and VRFY, are no longer supported:

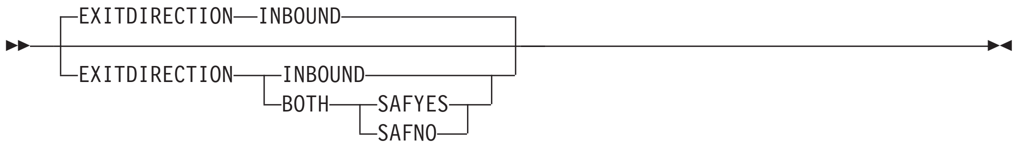
```
DISALLOWCMD   12
```

EXITDIRECTION statement

Use the EXITDIRECTION statement to control whether this SMTP calls the SMTP exit program provided by the customer to interrogate data being sent to SMTP from the JES spool.

Requirement: You must install a SMTP exit program for this function to work properly. For installation information for the SMTP exit program, see z/OS Communications Server: IP Configuration Guide and see “Steps for using the SMTP server exits” on page 1319.

Syntax



Parameters

INBOUND
Indicates that if an SMTP exit is installed by the customer, it is called *only* for inbound data from a TCP/IP connection. This is the default.

Tip: You do not need to code the EXITDIRECTION statement to invoke this behavior when an SMTP exit program is installed.

BOTH
Indicates that if an SMTP exit is installed by the customer, it is called for inbound data from a TCP/IP connection and for outbound data from the JES spool. The SMTP exit program uses the connection ID field (EZBPCNID) to determine where the data is coming from. EZBPCNID is always 257 for data coming from the JES spool.

Requirement: If BOTH is coded, you must code a second parameter (SAFYES/SAFNO). If a second parameter is not coded or coded incorrectly, INBOUND (the default) is used.

SAFYES
Indicates that for data coming only from the JES spool, SMTP requests SAF information from the JES interface. For more information about what is

provided in the SAF token, see z/OS Security Server RACF Data Areas, topic RUTKN Resource/User Security Token. The SAF token length is 80 bytes and the SAF token version is 1.

Even though a SAF token is requested, the SMTP exit must still be prepared to handle the possibility no SAF token being provided, in which case, the value of the SAF token address field (EZBPTOKP) is 0.

SAFNO
No SAF information is requested, therefore the SAF token address field (EZBPTOKP) is always 0.

Examples

In the following example, the SMTP exit program is called for data coming from both inbound TCP/IP connections and outbound JES connections. For the JES connection, SAF token information is requested.

```
EXITDIRECTION    BOTH SAFYES
```

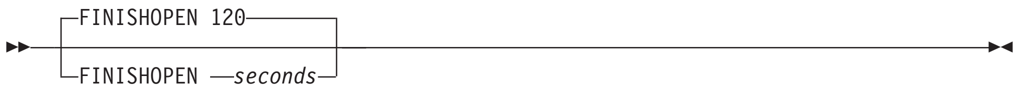
Usage notes

If the EXITDIRECTION statement is coded to support BOTH, recompile the SMTP exit program to include version 2 of the EZBZSMTP macro. Also, the SMTP exit program needs to handle the JES connection appropriately. For more information about exit programming, see “Steps for using the SMTP server exits” on page 1319.

FINISHOPEN statement

Use the FINISHOPEN statement to specify the number of seconds that SMTP waits while trying to establish a connection to a foreign site. After the specified number of seconds, SMTP ends the connection.

Syntax



Parameters

seconds
An integer in the range 1 - 86 400 indicating the number of seconds to wait for a connection to open. The default FINISHOPEN timeout is 120 seconds.

Examples

Set the timeout period to 90 seconds:

```
FINISHOPEN 90
```

GATEWAY statement

Use the GATEWAY statement to have SMTP operate as a mail gateway between TCP network sites and NJE network sites (if the host system is connected to both a TCP network and an NJE network).

Results:

- If you include the GATEWAY statement in the SMTP configuration data set, SMTP accepts mail addressed to users on NJE hosts defined in the data set pointed to by the //SMTPNJE DD statement in the SMTP cataloged procedure.
- If you do not specify GATEWAY, SMTP rejects all mail that arrives from the NJE network or host.

Syntax

```
►►──GATEWAY──────────────────────────────────────────────────────────────►◄
```

Parameters

There are no parameters for this statement.

Examples

You can configure the SMTP server with the GATEWAY statement to run as a mail gateway between TCP network users and users located on an NJE network attached to the local host. Figure 55 illustrates this configuration.

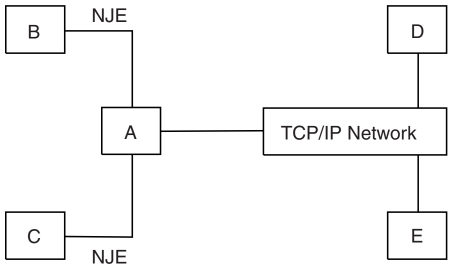


Figure 55. Example of a TCP-to-NJE mail gateway

In Figure 55:
- Host A is the local MVS host, running both TCP/IP and NJE.
- Hosts B and C are attached to host A through an NJE network.
- Hosts D and E are attached to host A through a TCP network.

Users on hosts A, B, and C can send mail or data sets to users on TCP hosts D and E using SMTPNOTE.

Usage notes

If you do not include the GATEWAY statement in the SMTP configuration data set, SMTP rejects all mail that arrives from NJE.

Related topics

- "LOCALCLASS statement" on page 1339
- "LOCALFORMAT statement" on page 1339
- "NJECLASS statement" on page 1346
- "NJEDOMAIN statement" on page 1346
- "NJEFORMAT statement" on page 1347

INACTIVE statement

Use the INACTIVE statement to specify the number of seconds of inactivity before SMTP considers a connection to be inactive and closes the connection.

Syntax

```
   ┌─INACTIVE 180──────┐
►►─┤                   ├──────────────────────────────────────────────►◄
   └─INACTIVE ──seconds┘
```

Parameters

seconds
: An integer in the range 1 - 86 400 that specifies the number of seconds after which SMTP considers the connection to be inactive. The default inactivity timeout is 180 seconds.

Examples

Use the following code to set the seconds of allowable inactivity to 90:

```
INACTIVE 90
```

INBOUNDOPENLIMIT statement

Use the INBOUNDOPENLIMIT statement to specify the maximum number of simultaneous TCP connections over which SMTP server receives mail. This number can be in the range 2 - 256 connections. The default, if this statement is not valid, is 256.

Guideline: This statement is optional. If it is not coded, the maximum number of TCP connections used by SMTP is limited to 256 (because it uses the PASCAL API).

Syntax

```
►►──INBOUNDOPENLIMIT──number──────────────────────────────────────────►◄
```

Parameters

number
: A value in the range 2 - 256 can be coded to reflect the maximum number of simultaneous TCP connections used by the SMTP server for inbound mail.

  Restriction:A value of 0 can be used and is a special case. If 0 is coded, then the SMTP server does not open a listening connection. Also, if 0 is coded, you cannot use AUTOLOG to monitor and restart the SMTP started procedure, because there is no listening connection to monitor. If this number is coded incorrectly, the default value of 256 is used.

Examples

Use the following code to set the maximum number of simultaneous TCP connections that are used by the SMTP server to 10:

```
INBOUNDOPENLIMIT   10
```

Usage notes

If 0 is coded, you cannot use AUTOLOG to monitor and restart SMTP.

Related topics

“OUTBOUNDOPENLIMIT statement” on page 1351

IPMAILERADDRESS statement

Use the IPMAILERADDRESS statement to reroute mail that was sent to an unknown host and direct it to an SMTP server on an IP network rather than to a user on a local or NJE network. The specified server should have network connectivity and be able to perform name resolution.

Results: The way IPMAILERADDRESS works depends on whether the RESOLVERUSAGE statement is coded.

- If RESOLVERUSAGE is Yes, then this statement only takes effect if the host name cannot be resolved (unknown host) using a domain name server specified in the *hlq*.TCPIP.DATA file or using search of the local hosts file.
- If RESOLVERUSAGE is No, all non-local mail destined for the IP network is forwarded to this IP address. Non-local mail is mail that has to go through an MTA (Mail Transfer Agent) to go to another host.

Syntax

```
►►──IPMAILERADDRESS──ip_address──────────────────────────────────────►◄
```

Parameters

ip_address
: The dotted decimal address of an SMTP server on an IP network.

Examples

In the following example, 7.89.250.72 is the address of the SMTP server on an IP network:

```
IPMAILERADDRESS 7.89.250.72
```

Usage notes

IPMAILERADDRESS, IPMAILERNAME and MAILER... UNKNOWN provide similar functions and cannot be used together.

Related topics

- “IPMAILERNAME statement” on page 1337
- See NSINTERADDR statement in the *hlq*.TCPIP.DATA file “MAILER statement” on page 1340. Also see z/OS Communications Server: IP Configuration Guide for information about sending messages to SMTP users and users on an IP network.
- z/OS Communications Server: IP Configuration Guide

- “RESOLVERUSAGE statement” on page 1355

IPMAILERNAME statement

Use the IPMAILERNAME statement to control whether this SMTP should forward non-local mail to an SMTP server on an IP network. The specified server should have network connectivity and be able to perform name resolution.

The SMTP code resolves the specified mailer name by doing an A query to the name servers (DNSs) configured in the TCPDATA data set (see “NSINTERADDR statement” on page 381) used by the SMTP started procedure. If no DNSs are configured, SMTP uses the local host tables to resolve the specified mailer name. SMTP does its own name resolution, normally using a UDP connection. It sends out the question (in this case an A query for the mailer's name) and interprets the name server's response. If the specified mailer name cannot be resolved, the mail is considered undeliverable.

Rule: For IPMAILERNAME, SMTP always uses a fully-qualified name for communication with the name server or for host table look-ups. Message EZA5645I in the SMTP output file, displays the fully-qualified name being used by SMTP.

For performance reasons, the location of the DNS is important to ensure timely responses to SMTP queries. The SMTP started procedure performs name resolution on every recipient even though it asks the same question to the name server. This allows the name server to change its response, and SMTP acts accordingly. However, after name resolution for the recipient is completed, SMTP uses the IP addresses associated with the recipient to send the mail.

The IPMAILERNAME statement has no defaults.

Requirement: You must specify a host name or a fully-qualified name for the SMTP mailer and an instruction indicating which mail to forward (ALL or UNKNOWN).

Syntax

```
►►──IPMAILERNAME──mailername──┬─ALL─────┬──────────────────────────────────►◄
                              └─UNKNOWN─┘
```

Parameters

mailername
: This represents the host name or the fully-qualified name (host.domain) of the IP mailer to which SMTP forwards the mail. If the host name is used, then SMTP appends the domain information collected from the TCPDATA data set (see “DOMAINORIGIN statement” on page 374). If the fully-qualified name is used, the name is not modified.

  Tip: A period (.) at the end of the name is considered a configuration error.

  SMTP does not check the validity of this name with regards to invalid characters or misspellings. The specified mailername is limited to 80 characters; this meets the SMTPCONFG data set limit.

ALL
: Indicates that SMTP should forward all non-local mail destined for the IP network to the specified IP mailer name.

UNKNOWN
Indicates that SMTP should forward only non-local mail destined for recipients on an unknown host.

Examples

Use the following code to cause SMTP to forward non-local mail destined for the IP network to your.mailer.name:

```
IPMAILERNAME  your.mailer.name  ALL
```

Long mailer names can be coded as follows:

```
IPMAILERNAME
this.is.a.very.long.mailer.name...
ALL
```

Usage notes

- IPMAILERADDRESS and MAILER... UNKNOWN provide similar functions and cannot be used when IPMAILERNAME has been coded.
- If RESOLVERUSAGE statement is coded it must be coded as RESOLVERUSAGE YES because this function requires the use of the DNS or local host tables.

Related topics

- "IPMAILERADDRESS statement" on page 1336
- "MAILER statement" on page 1340
- "RESOLVERUSAGE statement" on page 1355

LISTENONADDRESS statement

Use the LISTENONADDRESS statement to specify which IP address receives and sends mail on a multihomed system.

Syntax

```
►►──LISTENONADDRESS──ip_address──────────────────────────────────────────►◄
```

Parameters

ip_address
The dotted decimal address of an SMTP server on an IP network.

Examples

In the following example, 7.89.250.72 is the address of the SMTP server on an IP network that is the home address for mail:

```
LISTENONADDRESS 7.89.250.72
```

Related topics

"MAILER statement" on page 1340

LOCALCLASS statement

Use the LOCALCLASS statement to specify the spool class for local mail delivered by SMTP.

Syntax

```
    ┌─LOCALCLASS B─────────┐
►►──┼──────────────────────┼─────────────────────────────────────────────►◄
    └─LOCALCLASS ──class───┘
```

Parameters

class
The default is B (normally a punch class).

Examples

Use the following code to set the spool class for local mail delivered by SMTP:

```
LOCALCLASS B
```

Usage notes

The value used in this statement is site dependent. Before setting this class, check with your system administrator for the site-dependent information.

Guideline: Use the punch class of your system.

Related topics

- "GATEWAY statement" on page 1333
- "LOCALFORMAT statement"
- "NJECLASS statement" on page 1346
- "NJEDOMAIN statement" on page 1346
- "NJEFORMAT statement" on page 1347

LOCALFORMAT statement

Use the LOCALFORMAT statement to specify the spool data set format for mail delivered to users on the local host.

Syntax

```
    ┌─LOCALFORMAT NETDATA─┐
►►──┼─────────────────────┼──────────────────────────────────────────────►◄
    └─LOCALFORMAT PUNCH───┘
```

Parameters

NETDATA
For NETDATA format, records can be longer than 80 characters and arrive as message-type records. The data set name is the first eight characters of the sender's user ID.

NETDATA is the default format.

PUNCH
For PUNCH format, records are folded up to 80 characters in length or less. The spool data set is in NATIVE PUNCH format. The data set name is the first eight characters of the sender's user ID.

Examples

Use the following code to set the spool format for local mail delivered by SMTP:

```
LOCALFORMAT NETDATA
```

Usage notes

Use the default value of NETDATA, because the TSO RECEIVE command indicates that it has a file that is not valid with PUNCH format output.

Related topics

- "GATEWAY statement" on page 1333
- "LOCALCLASS statement" on page 1339
- "NJECLASS statement" on page 1346
- "NJEDOMAIN statement" on page 1346
- "NJEFORMAT statement" on page 1347

LOG statement

Use the LOG statement to log all SMTP traffic. The origin, sender, and recipients of each piece of mail are written to a log.

Syntax

```
►►──LOG──────────────────────────────────────────────────────────►◄
```

Parameters

There are no parameters for this statement.

Usage notes

The log information goes to the data set specified on the //LOGFILE DD statement of the SMTP cataloged procedure. If no //LOGFILE DD statement is included in the cataloged procedure, information is not logged.

If neither LOG or NOLOG is specified in the SMTP configuration data set, the default is LOG.

Related topics

"NOLOG statement" on page 1349

MAILER statement

Use the MAILER statement to specify the address of a batch SMTP server to which SMTP delivers mail destined for various classes of recipients.

Syntax

```
►►──MAILER──┬─user_id──────────┬──┬─PUNCH───┬──┬─SOURCEROUTES───┬──┬─LOCAL───┬──────►
            └─user_id@node_id──┘  └─NETDATA─┘  └─NOSOURCEROUTES─┘  └─NOLOCAL─┘

►──┬─NJE───┬──┬─UNKNOWN───┬──────────────────────────────────────────────────────►◄
   └─NONJE─┘  └─NOUNKNOWN─┘
```

Parameters

user_id
: Specifies the user ID of the local MAILER server.

user_id@node_id
: Specifies the NJE address of the MAILER server.

PUNCH
: Specifies that the MAILER server can only accept punch format spool data sets. Batch SMTP header records longer than 80 characters are split and an EBCDIC new-line character (hex 15) is placed in column 80 to indicate that the record is continued. Records within the body of the mail that are longer than 80 characters are split across multiple punch records.

NETDATA
: Specifies that the MAILER server accepts NETDATA format spool data sets. The NETDATA protocol automatically handles records longer than 80 characters.

SOURCEROUTES
: Specifies that the MAILER server accepts BSMTP header addresses with source routes.

  A source route contains routing information as well as the mailbox information. The following code is an example of a source route address:

  ```
  @host1,@host2:userid@host3.
  ```

  The mailbox information in this example is `userid@host3.`

NOSOURCEROUTES
: Specifies that the MAILER server does not accept source routes in the BSMTP header addresses.

  Specifying NOSOURCEROUTES indicates that the address strings must be mailbox information only.

LOCAL
: Specifies that mail for local recipients is spooled to the MAILER server.

NOLOCAL
: Specifies that mail for local recipients is spooled directly to the recipients.

NJE
: Specifies that mail for recipients on the NJE network is spooled to the MAILER server.

NONJE
: Specifies that mail for recipients on the NJE network is spooled directly to the recipients.

UNKNOWN
Specifies that mail for recipients on an unknown host is spooled to the MAILER server. This option is affected if the RESOLVERUSAGE statement is coded as No.

NOUNKNOWN
Specifies that mail for recipients on unknown hosts is returned to the sender as undeliverable.

Examples

Use the MAILER option if you run with the Columbia Mailer.

```
MAILER MUSER@MNODE PUNCH NOSOURCEROUTES LOCAL NJE UNKNOWN
```

Usage notes

- The MAILER server must either have a local address or be on the associated NJE network. The MAILER statement has no defaults; you must specify the parameters you want to use.
- IPMAILERADDRESS and MAILER... UNKNOWN provide the same function and should not be used together.
- All MAILER statement parameters must be specified or an error occurs and SMTP terminates; eliminating a parameter causes SMTP configuration to read the next statement in SMTPCONF as part of the mailer statement.

Related topics

- "IPMAILERADDRESS statement" on page 1336
- "RESOLVERUSAGE statement" on page 1355

MAILFILEDSPREFIX statement

Use the MAILFILEDSPREFIX statement to specify the prefix that is added to the SMTP mail data sets. If multiple MVS systems share the same volume for SMTP mail data sets, specify a unique prefix qualifier for each SMTP server on MAILFILEDSPREFIX.

Data sets created with this prefix contain mail that is in the process of being received or delivered. Each piece of mail queued for delivery occupies a minimum of 2 tracks.

Syntax

```
►►──MAILFILEDSPREFIX──prefix──────────────────────────────────────────►◄
```

Parameters

prefix
The prefix to add to the mail data sets. The prefix can be up to 20 characters in length, and a trailing period does not specified. The default is the name of the job running SMTP.

Examples

Set the prefix name for where incoming mail is stored while it is being queued for delivery:

```
MAILFILEDSPREFIX SMTP
```

Usage notes

All data sets are cataloged.

Related topics

- "MAILFILEUNIT statement"
- "MAILFILEVOLUME statement"

MAILFILEUNIT statement

Use the MAILFILEUNIT statement to specify the unit where the newly created SMTP mail data sets reside.

Syntax

```
      ┌─MAILFILEUNIT SYSDA──────────┐
►►────┼─────────────────────────────┼──────────────────────────────────────────►◄
      └─MAILFILEUNIT ──unit_name────┘
```

Parameters

unit_name
The unit name where the data sets reside. The default is SYSDA.

Examples

Use the following code to set the unit name for where incoming mail is stored while it is being queued for delivery:

```
MAILFILEUNIT SYSDA
```

Related topics

- "MAILFILEDSPREFIX statement" on page 1342
- "MAILFILEVOLUME statement"

MAILFILEVOLUME statement

Use the MAILFILEVOLUME statement to specify the volume where newly allocated SMTP mail data sets reside.

Syntax

```
►►──MAILFILEVOLUME──volume_name────────────────────────────────────────────────►◄
```

Parameters

volume_name
The volume name where the data sets reside. There is no default.

Examples

Set the volume name for where incoming mail is stored while it is being queued for delivery:

```
MAILFILEVOLUME volume6
```

Usage notes

- The SMTP volume selected must be able to accommodate the largest piece of mail (see “MAXMAILBYTES statement”). In addition, the VTOC indices need to be able to accommodate the number of mail pieces being processed.
- If the volume name is not specified, SMTP allocates a storage volume.
- If your system does not have storage volumes, you must specify a volume name.

Related topics

- “MAILFILEDSPREFIX statement” on page 1342
- “MAILFILEUNIT statement” on page 1343
- “MAXMAILBYTES statement”

MAXMAILBYTES statement

Use the MAXMAILBYTES statement to specify the maximum size in bytes of mail that is accepted over a TCP connection. Reply code of '552 Mail file too large' is sent to the remote SMTP client if the number of mail bytes arriving exceeds this value. This value is also used to determine the space allocation requirements for the data sets which hold the mail during processing (see Usage notes in this topic). These data sets names are &mailfiledsprefix.*..NOTE and occupy a minimum of 2 tracks per data set.

Syntax

```
    ┌─MAXMAILBYTES 524288──┐
►►──┼──────────────────────┼──────────────────────────────────────────────►◄
    └─MAXMAILBYTES ──bytes─┘
```

Parameters

bytes
: The maximum number of bytes for incoming or outgoing mail. Mail arriving that is larger than this size, over a TCP connection, is rejected. The limits for this statement are 1 - 2 147 483 647. The default size is 524 288 (512KB) bytes.

Examples

Set the maximum size for mail to 32KB:

```
MAXMAILBYTES 32768
```

Usage notes

- Note that the spool volume must be able to accommodate the number of bytes set in MAXMAILBYTES.
- The value used for *bytes* in the MAXMAILBYTES statement determines the space allocations for data sets allocated to hold the mail while it is being processed

and is waiting for delivery. Be careful not to use too large a value, or the data sets allocated are too large. This can vary with configuration of DF/SMS on your system. However, in general, the allocation used for the NOTE data sets is equivalent to specifying SPACE=(6233,(aaaaa,bbbb)), where aaaa=round((MAXMAILBYTES/4000)+1) and bbbb=round(aaaa/2).

Related topics

- “CHECKSPOOLSIZE statement” on page 1327
- “MAILFILEDSPREFIX statement” on page 1342
- “MAILFILEVOLUME statement” on page 1343

MAXMSGSENT statement

Use the MAXMSGSENT statement to control the number of messages to be sent out on a single TCP/IP connection by the SMTP client. This statement is optional; if it is not coded, it defaults to the value of 0, which indicates that no limit is set.

If the statement is coded, the SMTP client closes the current connection when the maximum number of messages has been sent. If there is more mail to send, then the mail is returned to the active queue where eventually a new connection is opened. This statement effects performance; the fewer messages that can be sent, the slower the mail processing.

Guideline: The limit that is set is used for all outbound TCP/IP connections. Also, if the DEBUG statement is enabled, the SYSDEBUG log indicates when a connection is closed due to the maximum number of messages sent.

This statement does not affect the number of messages processed on the batch connection (257).

Syntax

```
   ┌─MAXMSGSENT 0──┐
►►─┤               ├──────────────────────────────────────────────────────────►◄
   └─MAXMSGSENT nn─┘
```

Parameters

nn A decimal number in the range 0 - 2 147 483 647 (MAXINT). Coding the value 0 indicates that there is no limit on the number of messages to send.

A number greater than 2 147 483 647 fails with the error message `EZA5649E Invalid MaxMsgSent value: <nn>`. This occurs because the value is too large to be stored in a signed 32-bit integer.

The default MAXMSGSENT value is 0, which indicates no limit.

Examples

Set the maximum number of messages to send on a single TCP/IP connection to 1000:

```
MAXMSGSENT 1000
```

Usage notes

Certain vendor servers might limit the number of messages that they can receive. Setting this parameter to match the vendor server can assist in avoiding undeliverable mail.

NJECLASS statement

Use the NJECLASS statement to specify the spool class for mail delivered by SMTP to the NJE network.

Syntax

```
         ┌─NJECLASS B────────┐
►►───────┤                   ├──────────────────────────────────────────────►◄
         └─NJECLASS ──class──┘
```

Parameters

class
: The spool class for mail delivered by SMTP. The default is B (which is normally a punch class).

Examples

Set the spool class for mail delivered to B:

```
NJECLASS B
```

Usage notes

This statement is site-dependent. Before setting the class, check with your JES system administrator for site-dependent information. The preferred setting is the punch class of your system.

Related topics

- "GATEWAY statement" on page 1333
- "LOCALCLASS statement" on page 1339
- "LOCALFORMAT statement" on page 1339
- "NJEDOMAIN statement"
- "NJEFORMAT statement" on page 1347

NJEDOMAIN statement

Use the NJEDOMAIN statement to specify the domain name of the NJE network when SMTP is running as a mail gateway for other NJE or RSCS hosts.

Guideline: This is an optional statement and is not needed in all gateway situations.

The term NJEDOMAIN is a unique concept designed for z/OS SMTP that can act as a mail gateway between the NJE and the TCP/IP networks. Basically, the NJEDOMAIN is just a name so that mail addressing strings can have the form: NJEhost.NJEdomain. When the domain portion of the mail addressing string matches the NJEDOMAIN name, SMTP treats the associated host as an NJE host; therefore, SMTP does not try to resolve the host's name to an IP address. SMTP checks the NJE host against a list of NJE hosts contained in the data set associated with the DD card SMTPNJE in SMTP started task JCL. If this host is not in the list of NJE hosts, it is considered an unknown host.

Guideline: The NJEDOMAIN name can contain a dot.

The NJE domain name is also used in the default set of rewrite rules for the RFC 822 header fields of mail passing from NJE network senders to TCP/IP network or NJE network recipients. For more information about the RFC 822 rewrite rules and whether they are enabled, see z/OS Communications Server: IP Configuration Guide and see “REWRITE822HEADER statement” on page 1358.

Syntax

```
►►──┬──────────────────────────────┬──────────────────────────────────────►◄
    └─NJEDOMAIN──njedomain_name──┘
```

Parameters

njedomain_name
The NJE domain name. The default is a null string.

Examples

Use the following code to set the NJE domain to BITNET:

```
NJEDOMAIN BITNET
```

Usage notes

SMTP considers the NJE domain name BITNET to be a synonym for the European Academic Research Network (EARN and EARNET). This statement can affect the local processing of mail regardless of whether the GATEWAY statement is coded or not.

Related topics

- “GATEWAY statement” on page 1333
- “LOCALCLASS statement” on page 1339
- “LOCALFORMAT statement” on page 1339
- “NJECLASS statement” on page 1346
- “NJEFORMAT statement”
- “REWRITE822HEADER statement” on page 1358

NJEFORMAT statement

Use the NJEFORMAT statement to specify the spool data set format for mail delivered to recipients on the NJE network.

Syntax

```
   ┌─NJEFORMAT NETDATA──────────┐
►►─┼────────────────────────────┼──────────────────────────────────────────►◄
   └─NJEFORMAT──┬─PUNCH───┬─────┘
                └─NETDATA─┘
```

Parameters

PUNCH
: Specifies that records are folded to 80 characters in length or fewer.

NETDATA
: Specifies that records can be longer than 80 characters and that they arrive as MESSAGE-type records. The default format is NETDATA.

Examples

Use the following code to set the format in which mail is sent to NJE recipients to PUNCH:

```
NJEFORMAT PUNCH
```

Usage notes

This statement is valid only in GATEWAY mode.

Related topics

- "GATEWAY statement" on page 1333
- "LOCALCLASS statement" on page 1339
- "LOCALFORMAT statement" on page 1339
- "NJECLASS statement" on page 1346
- "NJEDOMAIN statement" on page 1346

NJENODENAME statement

Use the NJENODENAME statement to specify the NJE node name of the local JES2 or JES3 node for SMTP. This statement overrides the value in the IEFSSN member and is an alternative to forcing the users to specify their real NJE node name in the IEFSSN member. It also allows users to easily correct the name for SMTP's use, in case it was spelled wrong. Previously, users were required re-IPL to change the member because it was the only place from which SMTP would get the NJE node name. This value is not used in the place of the IEFSSN value as a selector in TCPIP.DATA.

Guideline: NJENODENAME is location sensitive.

Syntax

```
►►──NJENODENAME──njenode_name──────────────────────────────────────────►◄
```

Parameters

njenode_name
: The NJE node name of the local JES2 or JES3 node. The default is a null string.

Examples

Use the following code to set the NJE node name to ALMADEN:

```
NJENODENAME ALMADEN
```

Usage notes

The NJENODENAME statement, if specified, must precede any of the following statements in the SMTP Configuration Data Set:

- ALTNJEDOMAIN
- MAILER
- NJEDOMAIN
- SMSGAUTHLIST

Related topics

- "NJECLASS statement" on page 1346
- "NJEDOMAIN statement" on page 1346
- "NJEFORMAT statement" on page 1347

NOLOG statement

Use the NOLOG statement to turn off logging information that indicates that mail has been received and delivered.

Syntax

```
►►──NOLOG──────────────────────────────────────────────────────────────►◄
```

Parameters

There are no parameters for this statement.

Usage notes

If neither LOG or NOLOG is specified in the SMTP configuration data set, the default is LOG.

Related topics

"LOG statement" on page 1340

NOSOURCEROUTE statement

Use the NOSOURCEROUTE statement to control whether this SMTP generates and passes a source routing string for the originator address (MAILCMD) or for the recipient address (RCPTCMD). Setting the parameter to ENABLED causes no source routing addresses to be generated for both the MAIL FROM: and RCPT TO: SMTP commands. A source route is a path that contains a source routing list of hosts and a destination mailbox. The list of hosts is the route information. It describes how the mail is to arrive at its final destination. The mail is passed from one host in the list to the next until it is delivered to the intended recipient.

Source routing addressing string has the following format:

```
@host1,@host2,@host3:myuserid@myhost
```

where myuserid@myhost is considered the mailbox information.

NOSOURCEROUTE DISABLED is the default and indicates that source route on the MAIL FROM: and RCPT TO: commands are honored.

Syntax

```
                    ┌─DISABLED─┐
►►──NOSOURCEROUTE───┼──────────┼──────────────────────────────────────►◄
                    ├─MAILCMD──┤
                    ├─RCPTCMD──┤
                    └─ENABLED──┘
```

Parameters

DISABLED
Source routing address strings are generated for both the MAIL FROM: and the RCPT TO: SMTP commands based on the source routing rules documented in RFC 821. This is the default if nothing is specified, or if what is specified is not a valid parameter.

MAILCMD
Source routing address strings are *not* generated for the MAIL FROM: SMTP command. This means that the return path only contains the mailbox information. However, the RCPT TO: SMTP command maintains source routing addressing based on the source routing rules documented in RFC 821.

RCPTCMD
Source routing address strings are *not* generated for the RCPT TO: SMTP command. This means that the send path only contains the mailbox information. However, the MAIL FROM: SMTP command maintains source routing addressing based on the source routing rules documented in RFC 821.

ENABLED
Source routing address strings are *not* generated for both the MAIL FROM: and the RCPT TO: SMTP commands. Only the mailbox information is provided.

Examples

To stop SMTP from adding its host identifier in front of the mailbox information about the return path passed on the MAIL FROM: SMTP command, code the following statement:

```
NOSOURCEROUTE   MAILCMD
```

Usage notes

- The removal of these source routes might make the return path unusable. This is a potential problem when the originating host is not directly accessible to any mail transfer agent that must generate error mail to the originating host.
- The removal of these source routes might make understanding of which route is used to deliver the mail difficult for debugging situations. However, if the delivered piece of mail can be viewed, then receive lines can be used instead.

- Only one NOSOURCEROUTE statement should be coded. If there is more than one NOSOURCEROUTE statement in the SMTPCONF data set, then only the last statement is used.

OUTBOUNDOPENLIMIT statement

Use the OUTBOUNDOPENLIMIT statement to specify a limit on the maximum number of simultaneous TCP connections over which SMTP can actively deliver mail. The OUTBOUNDOPENLIMIT statement should only be used if there are limited TCP resources on the system and SMTP is using too many of these resources.

SMTP is a PASCAL application and is limited to a total of 256 simultaneous connections due to the PASCAL API.

Syntax

```
►►──OUTBOUNDOPENLIMIT──number_of_connections──────────────────────────────►◄
```

Parameters

number_of_connections
The maximum number of simultaneous connections. This number must be in the range 1 - 256.

Examples

Use the following code to set the maximum number of simultaneous TCP connections to which mail is sent to 100:

```
OUTBOUNDOPENLIMIT 100
```

PORT statement

Use the PORT statement to control the local port used by the SMTP server for receiving incoming mail. This statement is optional; if this statement is not coded, it defaults to the value 25, which is the well known port for mail. The port number 25 is typically reserved (in *hlq*.PROFILE.TCPIP) for the SMTP server to accept incoming mail request. If another port number is selected for the SMTP server, then update the hlq.PROFILE.TCPIP file accordingly.

Tip: Avoid using a number in the well-known ports range (1 - 1023), which can be reserved for other servers.

If the statement is coded, the SMTP server uses this port value to open a listening port for incoming mail. On the z/OS platform, the corresponding configuration statement that needs to be modified for the SMTP client is the REMOTEPORT statement in the SMTP configuration file on the system from where the SMTP client is sending the mail.

Syntax

```
     ┌─PORT 25────────┐
►►───┼────────────────┼─────────────────────────────────────────────────►◄
     └─PORT ──port_num┘
```

Parameters

port_num
: An integer in the range 1 - 65 535 that specifies the port number to which SMTP listens. This parameter is limited to ten characters.

  Requirement: The *port_num* value must be in the range 1 - 65 534. If you specify a value outside this range, SMTP is not started.

Examples

Use the following code to set the port for incoming mail to port 2 000:

```
PORT 2000
```

Usage notes

- You can specify a port number only if it has not already been reserved for some other server in *hlq*.PROFILE.TCPIP.
- This statement is for system testing only.

Related topics

- “REMOTEPORT statement” on page 1354
- SHAREPORT parameter description in “PORT statement” on page 257

POSTMASTER statement

Use the POSTMASTER statement to specify the user ID to which SMTP delivers all mail addressed to POSTMASTER.

Syntax

```
    ┌─POSTMASTER TCPMAINT─────────────┐
►►──┴─POSTMASTER──┬─user_id─────────┬─┴──────────────────────────────────►◄
                  └─user_id@node_id─┘
```

Parameters

user_id
: The user ID on the local system to which mail addressed to POSTMASTER should be delivered. The default ID is TCPMAINT.

user_id@node_id
: The NJE or SMTP address to which mail addressed to POSTMASTER should be delivered.

Examples

Use the following code to set the user ID to receive the POSTMASTER mail to MAILGUY at POSTOFC:

```
POSTMASTER MAILGUY@POSTOFC
a.b.c@company.com
mail.guy@postofc.com
```

Usage notes

- To specify multiple recipients to receive mail addressed to POSTMASTER, code a separate POSTMASTER statement for each recipient. There is no limit on the number of POSTMASTER statements that you can code.
- In SECURE mode, you can only specify the POSTMASTER statement once for a local user ID as the single recipient of mail addressed to POSTMASTER.

Related topics

“SECURE statement” on page 1359

RCPTREPLY452 statement

Use the RCPTREPLY452 statement to cause the SMTP client to interpret the 452 reply code being received on the RCPT command as 'too many recipients'. Note that the SMTP client continues to handle the 552 reply code as 'too many recipients' also.

If this statement is not present, the SMTP client interprets the 452 reply code as 'insufficient system storage', and the mail processing is halted.

Syntax

```
►►──RCPTREPLY452──────────────────────────────────────────────────────►◄
```

Parameters

None

Examples

Use the following code to change the SMTP client's interpretation of the 452 reply code received on a RCPT command:

```
RCPTREPLY452
```

RCPTRESPONSEDELAY statement

Use the RCPTRESPONSEDELAY statement to specify how long the SMTP server delays responding to the RCPT commands from the sender SMTP, while it is waiting for domain name resolution.

Syntax

```
     ┌─RCPTRESPONSEDELAY 60────────┐
►►───┴─RCPTRESPONSEDELAY ──seconds─┴──────────────────────────────────►◄
```

Parameters

seconds
: A number in the range 0 - 86 400 specifying the number of seconds SMTP waits before responding to the RCPT TO command. The default is 60 seconds.

Examples

Use the following code to set the RCPT TO: response time to 90 seconds:

```
RCPTRESPONSEDELAY 90
```

Usage notes

If resolution does not complete before the specified period, the SMTP server assumes name resolution is successful and does the following tasks:

- Sends the following message to the sender SMTP: `250 ok`.
- Queues the recipient address for asynchronous resolution.

If SMTP later determines that the recipient address cannot be resolved, the mail is returned to the sender.

Related topics

"RESOLVERRETRYINT statement" on page 1355

REMOTEPORT statement

Use the REMOTEPORT statement to control which remote port number the SMTP client uses for sending outbound mail. This statement is optional; if it is not coded, it defaults to the value 25, which is the well known port for mail.

If you code this statement, then the SMTP client uses this port value to connect to the remote SMTP server. If no SMTP server is listening on that port, then mail cannot be delivered. On the z/OS system, ensure that the value configured for the PORT statement in the configuration file for the SMTP server is the same as the value configured on the REMOTEPORT statement in the configuration file for the SMTP client.

You can use this statement for system testing of the SMTP function.

Syntax

```
      ┌─REMOTEPORT 25──────┐
►►────┼────────────────────┼──────────────────────────────────────────────►◄
      └─REMOTEPORT ──nn────┘
```

Parameters

nn The *nn* value is a decimal number. This parameter must be within the range 1- 65 534 and is limited to ten characters.

Requirement: If the REMOTEPORT statement is coded, it must be within the range 1- 65 534. No default taken if the statement is coded incorrectly, and SMTP does not start.

Examples

Use the following code to set the remote port for outbound mail sent by the SMTP client to port 2000:

```
REMOTEPORT 2000
```

Related topics

"PORT statement" on page 1351

RESOLVERRETRYINT statement

Use the RESOLVERRETRYINT statement to specify the number of minutes SMTP waits between attempts to resolve domain names.

Syntax

```
   ┌─RESOLVERRETRYINT 20───────┐
►►─┼───────────────────────────┼──────────────────────────────────────►◄
   └─RESOLVERRETRYINT ──minutes┘
```

Parameters

`minutes`
: A number in the range 1 - 1 439 specifying the number of minutes between each attempt to resolve a domain name if the name server is causing delays. The default is to try resolution again every 20 minutes.

Examples

Use the following code to set the waiting time between attempts to resolve domain names to 30 minutes:

```
RESOLVERRETRYINT 30
```

RESOLVERUSAGE statement

Use the RESOLVERUSAGE statement to indicate whether SMTP should attempt to resolve non-local domain names.

Syntax

```
   ┌─RESOLVERUSAGE YES─┐
►►─┼───────────────────┼──────────────────────────────────────────────►◄
   └─RESOLVERUSAGE NO──┘
```

Parameters

YES
: Specifies that SMTP should attempt normal domain name resolution. This is the default.

NO Specifies that SMTP should not attempt to resolve any non-local domain names using the DNS or local host tables. Any mail received by SMTP that is addressed to non-local domain names is considered unknown.

Examples

In the following example, the IPMAILERADDRESS statement is being used in conjunction with the RESOLVERUSAGE statement to forward all non-local mail to IP address 1.2.3.4, where another SMTP server resides:

```
IPMAILERADDRESS 1.2.3.4
RESOLVERUSAGE NO
```

Usage notes

- If IPMAILERNAME and RESOLVERUSAGE statements are coded, RESOLVERUSAGE YES must be used.
- This statement should only be specified when you want to configure SMTP to send all non-local mail to a specified mail server, or mail relay. You might need to do this if you have installed a firewall. As a result, if you specify NO on this statement, you should specify a target mail server using the IPMAILERADDRESS or MAILER...UNKNOWN statement. Non-local mail is mail that has to go through an MTA (Mail Transfer Agent) to go to another host.

  Restriction: You can only specify one of these statements.
- If you specify RESOLVERUSAGE NO and do not specify an IPMAILERADDRESS or MAILER ... UNKNOWN statement, a warning message is issued during SMTP initialization, and all non-local mail is returned to the sender as undeliverable.

Related topics

- “IPMAILERADDRESS statement” on page 1336
- “IPMAILERNAME statement” on page 1337
- “MAILER statement” on page 1340

RESTRICT statement

Use the RESTRICT statement to specify addresses of users who cannot use SMTP services. This includes sending and receiving mail.

Syntax

```
                                   ┌──────────┐
                                   ▼          │
►►──RESTRICT──┬─PURGE──────┬────userid──┴──ENDRESTRICT─────────────────────────►◄
              ├─RETURN─────┤
              └─TRANSFERTO─┘
```

Parameters

PURGE
: Specifies that the spool data set is to be purged.

RETURN
: Specifies that the spool data set is to be returned to the originator.

TRANSFERTO
: Specifies that the spool data set is to be forwarded to the specified *userid*.

userid
: Specifies the address of the user.

Examples

In the following example, mail from restricted users is returned, no mail is accepted from KNIGHT at 2 different nodes, and no mail is accepted from anyone on the host CASTLE:

```
RESTRICT RETURN
  KNIGHT@CAMPTENT
  KNIGHT@TOURNMNT
  *@CASTLE
ENDRESTRICT
```

Usage notes

- You can use a wildcard character (*) in the user identifier string, or the host/domain identifier string, or both. These two strings are separated by the @ character (for example, userid@hostid). It can be used to replace the entire string. For example, *@castle restricts all the users at castle. You can use it to replace a portion of the string when it is appended to the end of the string. For example, mary*@castle restricts all the user IDs beginning with the character string mary at castle. However, the wildcard character cannot be used as a prefix to a string or embedded within the string.
- Specifying *@* causes no mail to be sent or accepted, and results in `undeliverable mail` messages to be issued.
- The ENDRESTRICT statement ends the RESTRICT statement.
- If SMTP receives a spool data set from a restricted user, the spool data set is as follows:
  - Purged, if PURGE is specified
  - Returned to the originator, if RETURN is specified
  - Forwarded to a specific user ID, if TRANSFERTO is specified

  In addition, SMTP rejects any MAIL FROM or RCPT TO commands whose destinations are restricted users.
- The TCPIP and NJE address must be included in the RESTRICT statement list in order to restrict a user from sending and receiving mail. SMTP rejects only addresses that are in the restrict list; it does not check for aliases. For example, you can restrict `user@host1`. If `host2` is an alias for `host1`, mail for `user@host2` is not rejected unless `user@host2` is also in the restrict list.
- When the RESTRICT statement is used, incoming mail must be in NETDATA format.
- The RESTRICT statement cannot be used if the SMTP server is running as a secure gateway. Either remove or comment out the RESTRICT statements from the SMTP configuration data set.
- The RESTRICT statement cannot be used in combination with the SECURE statement.

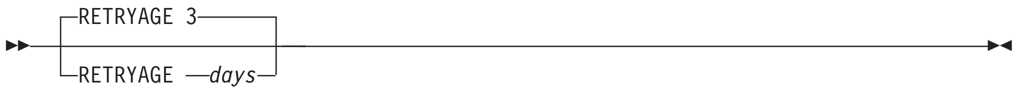
Related topics

- “LOCALFORMAT statement” on page 1339
- “MAILER statement” on page 1340
- “NJEFORMAT statement” on page 1347
- “SECURE statement” on page 1359

RETRYAGE statement

Use the RETRYAGE statement to specify the number of days after which SMTP returns mail as undeliverable. SMTP tries to deliver mail to an inactive site. After the number of days specified on this statement, SMTP returns the mail to the sender with a note listing any recipients to which the mail could not be delivered.

Syntax



Parameters

days
: A number in the range 1 - 365 specifying the number of days to try to deliver the mail. The default is for SMTP to try to deliver a piece of mail for 3 days before returning it.

Examples

Use the following code to continue trying to deliver mail for 2 days:

```
RETRYAGE 2
```

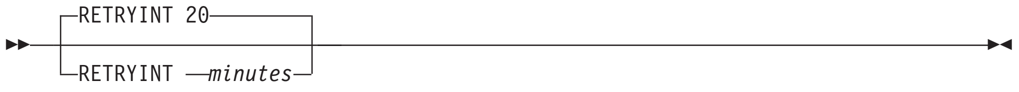
Related topics

"WARNINGAGE statement" on page 1364

RETRYINT statement

Use the RETRYINT statement to specify the number of minutes SMTP should wait between attempts to deliver mail to an inactive host.

Syntax



Parameters

minutes
: A number in the range 1 - 1 439 specifying the number of minutes between each attempt to deliver the mail. The default is to try to establish a connection to these sites every 20 minutes.

Examples

Use the following code to continue to try to redeliver mail every 30 minutes:

```
RETRYINT 30
```

REWRITE822HEADER statement

Use the REWRITE822HEADER statement to specify whether SMTP should rewrite or print the RFC 822 headers of mail arriving from the NJE side of the mail gateway.

Syntax

```
     ┌─REWRITE822HEADER YES NOPRINT────────────┐
►►───┼─────────────────────────────────────────┼──────────────────────────────────►◄
     └─REWRITE822HEADER──┬─NO──────────────────┤
                         │         ┌─NOPRINT─┐ │
                         └─YES─────┼─────────┼─┘
                                   └─PRINT───┘
```

Parameters

NO Specifies that SMTP should not rewrite the RFC 822 mail headers.

Guideline: Do not use this unless all mail user agents sending mail to SMTP create RFC 822 mail headers with fully qualified domain addresses that are valid on the Internet.

NOPRINT
Specifies that SMTP should not print the RFC 822 header rewriting rules to the console when SMTP starts.

PRINT
Specifies that SMTP should print the RFC 822 header rewriting rules to the console when SMTP starts.

YES
Specifies that SMTP should rewrite the RFC 822 mail headers. The YES parameter with NOPRINT is the default. SMTP uses a set of default header rewriting rules.

Examples

Use the following code to rewrite the RFC 822 headers on all mail passing from NJE to TCP through the mail gateway and print the rules to the SMTP output when SMTP starts:

```
REWRITE822HEADER YES PRINT
```

Usage notes

The SMTP.RULES data set specifies how the server is to rewrite the headers.

Related topics

- “GATEWAY statement” on page 1333

SECURE statement

Use the SECURE statement to specify that SMTP operates as a secure mail gateway between TCP network sites and NJE network sites.

Syntax

```
►►──SECURE──────────────────────────────────────────────────────────────────────►◄
```

Parameters

There are no parameters for this statement.

Usage notes

- Do not use the GLOBALTCPIPDATA file and SECURE, if you want to use a different domain origin than what is specified in the GLOBALTCPIPDATA file. The domain origin that is specified in the GLOBALTCPIPDATA file is used in the MAIL FROM statement.
- The SECURE statement cannot be used in combination with the RESTRICT statement.
- Mail is accepted through the secure gateway only if the NJE user IDs and node IDs are included in the SMTP security table (SMTP security data set).
- When the SECURE statement is used, mail must be in NETDATA format.
- If you specify the SECURE statement, then source routing is disabled to prevent the gateway from relaying mail to unauthorized users.

The data set pointed to by the //SECMEMO DD statement in the SECTABLE data set is sent to NJE users that are not authorized to use the gateway.

Related topics

- “LOCALFORMAT statement” on page 1339
- “MAILER statement” on page 1340
- “NJEFORMAT statement” on page 1347
- “RESTRICT statement” on page 1356

SMSGAUTHLIST statement

Use the SMSGAUTHLIST statement to specify the local users authorized to issue privileged SMTP SMSG commands.

Restriction: Any TSO user can issue the general usage SMTP SMSG commands, but only those users specified in the SMSGAUTHLIST statement can issue the privileged commands.

Privileged SMTP SMSG commands allow the shutting down of SMTP and the enabling or disabling of various SMTP trace and debug options.

Syntax

```
                  ┌────────────┐
                  ▼            │
►►──SMSGAUTHLIST────user_id────┴──ENDSMSGAUTHLIST──────────────────────────►◄
```

Parameters

user_id
: Specifies the address of a local user ID authorized to issue privileged SMTP SMSG commands. The *user_id* parameter can be repeated.

Examples

Specify the local users authorized to issue privileged SMTP SMSG commands:

```
SMSGAUTHLIST
    TCPMAINT
    OPERATOR CHANCE
ENDSMSGAUTHLIST
```

Usage notes

- The ENDSMSGAUTHLIST statement ends the SMSGAUTHLIST statement.
- You must add users to this statement only if you are invoking privileged SMSG commands from the TSO command line. A user issuing the MODIFY *smtpprocname*, SMSG command from the system console is considered to be authorized; that user's user ID does not need to be added to the SMSGAUTHLIST statement.

Related topics

See z/OS Communications Server: IP User's Guide and Commands for more information.

SPOOLPOLLINTERVAL statement

Use the SPOOLPOLLINTERVAL statement to specify the interval (in seconds) for SMTP to check the spool for incoming batch data sets.

Syntax

```
►►──SPOOLPOLLINTERVAL──seconds──────────────────────────────────────────────►◄
```

Parameters

seconds
The number of seconds between each check. The range is 5 - 3 600 seconds (3 600 seconds equals one hour).

Examples

Set the time between spool polling to 30 seconds:

```
SPOOLPOLLINTERVAL 30
```

Usage notes

If the value for *seconds* is too low, system overhead is increased; if the value is too high, incoming mail must wait to be processed.

STOPONRENF statement

Use the STOPONRENF statement to control the behavior of the SMTP server so if a RENAME failure occurs on a data set associated with the batch connection (257), then the SMTP server stops by normal termination.

Some RENAME problems are recoverable by SMTP; if the RENAME problem occurs while processing a TCP connection, the remote SMTP is responsible for resending the mail. However, if mail is being processed from the batch connection 257 (JES spool), a note might be lost.

Currently, SMTP generates error message EZA5544E to the system console to indicate that a RENAME failure occurred. If the data set involved is *smtphlq*.CONN257.NOTE, this is the batch connection. If the RENAME failure is persistent, batch jobs might need to be rerun in order to recover the mail. The system administrator should find out why RENAME is failing and correct the situation as soon as possible. Check the system console log for messages prior to the EZA5544E message that contain the failing data set name. Message EZA5391E is generated to the system console if this statement is coded and a RENAME failure on a data set associated with the batch connection occurs.

This option controls how SMTP behaves. By default, SMTP continues trying to process other notes.

Syntax

```
►►──STOPONRENF──────────────────────────────────────────────────────►◄
```

Parameters

None

Examples

Use the following code to cause the SMTP server to stop when a RENAME failure occurs on a data set associated with the batch connection:

```
STOPONRENF
```

TEMPERRORRETRIES statement

Use the TEMPERRORRETRIES statement to specify the number of times SMTP tries to redeliver mail to a host with a temporary problem. Temporary problems include network congestion, network connectivity, or a remote mail server responding with a reply code of 4xx indicating a temporary error condition when a 5xx reply code is appropriate. In this case, a single piece of mail may be blocking other mail in the queue from being processed due to the remote mail server reply code. Note that when using this statement mail can be returned to the originator sooner than expected because the number of retries have exceeded the value set by this statement.

Syntax

```
      ┌─TEMPERRORRETRIES 0────────┐
►►────┼───────────────────────────┼─────────────────────────────────►◄
      └─TEMPERRORRETRIES ──retries┘
```

Parameters

retries
: The number of times mail delivery to a host with a temporary problem is tried again. The default is 0.

Examples

Use the following code to attempt redelivery 5 times in cases where a temporary problem with the host to which mail has been addressed:

```
TEMPERRORRETRIES 5
```

Usage notes

- If delivery is still unsuccessful, the mail is returned to the sender.

- Change the number of *retries* from the default of 0 only when remote mail servers repeatedly terminate abnormally or hang SMTP mail transactions.
- If *retries* is 0 and there is a problem with the remote mail server, SMTP continues trying again to deliver the same piece of mail until it times out. The other mail sitting behind it in the queue waits for delivery until SMTP times out. This timeout is controlled by a value associated with the RETRYAGE statement.

TIMEZONE statement

Use the TIMEZONE statement to specify the printable name of the local time zone. If the printable name is SYSTZ, SMTPPROC gets the TIMEZONE value from the MVS system using the local TIME/DATE offset in the CVT (communication vector table) associated with SMTPPROC (SMTP started task). The local TIME/DATE offset is controlled by the System Administrator who sets the MVS system time/date and timezone parameters. See z/OS MVS Initialization and Tuning Guide and z/OS MVS System Commands for more information about the CLOCKxx parmlib member, the MVS SET CLOCK=hh.mm.ss command and the MVS SET TIMEZONE={W|E}.hh.mm command.

Syntax

```
     ┌─TIMEZONE EST──────────┐
►►───┼───────────────────────┼──────────────────────────────────────────────►◄
     └─TIMEZONE ──time_zone──┘
```

Parameters

time_zone
The name of the local time zone.

Requirement: This parameter must be a continuous character string in the range 1-5 characters. See RFC 822 for the valid character formats for this parameter. The default TIMEZONE value is EST.

The SMTP code does not check the validity of this parameter. If the parameter is set to SYSTZ, each time SMTP generates a new RCF 822 date/time and timezone header, it uses the value from the MVS system. The value is converted to a string format of plus (+) or minus (-) and 4 digits for example, -HHMM. This string is appended to the RFC 822 date/time header. See Appendix C, “Related protocol specifications,” on page 1465 for information about accessing RFCs.

Examples

To set the time zone to pacific standard time (PST), use the following code:

```
TIMEZONE PST
```

To set the time zone to local differential hours and minutes HHMM, use the following code:

```
TIMEZONE +1200
```

To set the time zone to the value used by the MVS system for local time/date offset, use the following code:

```
TIMEZONE SYSTZ
```

Usage notes

SMTPPROC does not alter any existing date/time stamp and timezone headers in the mail.

WARNINGAGE statement

Use the WARNINGAGE statement to specify the number of days after which a copy of the mail is returned to the sender with a warning. The warning is included in the header in the copy of the mail. It includes the following information:

- SMTP has been unable to deliver the mail thus far
- How many days the mail has been undeliverable
- How many days that SMTP continues to try to deliver the mail (derived from the RETRYAGE statement)

Syntax

```
   ┌─WARNINGAGE 3──────┐
►►─┤                   ├──────────────────────────────────────────────►◄
   └─WARNINGAGE ──days─┘
```

Parameters

days
: A number from 0 - 365 specifying the number of days to attempt delivery of the mail before sending a nondelivery warning to the sender. The default is 3 (same as the default for the RETRYAGE statement).

Examples

Use the following code to warn the sender that mail has been undeliverable for one day, but that SMTP continues to attempt delivery for another two days, code the following:

```
RETRYAGE 3
WARNINGAGE 1
```

Usage notes

SMTP only sends a warning if the number of days specified on the WARNINGAGE statement is less than the number of days specified on the RETRYAGE statement. When the number of days specified by the WARNINGAGE statement is greater than or equal to the number of days specified on the RETRYAGE statement, no warning is issued to the sender.

Related topics

"RETRYAGE statement" on page 1357

Chapter 30. Communications Server SMTP application

The Communications Server SMTP (CSSMTP) application sends mail messages from a JES spool data set to an SMTP server.

For additional overview and configuration information about CSSMTP, see the information about the Communications Server SMTP (CSSMTP) application in z/OS Communications Server: IP Configuration Guide.

This topic contains the following information:

General syntax rules for CSSMTP

The following list shows the general syntax rules for CSSMTP:

- Specify CSSMTP configuration files using the code page set in the environment variable CSSMTP_CODEPAGE_CONFIG or use default value IBM-1047 for EBCDIC.
- Each statement must have a corresponding value and must be separated from its value by one or more blank spaces.
- Only one attribute and its value can be specified per line.
- Text beyond the specified attribute and value is ignored.
- If the first non-blank character on a line begins with the number sign (#), then the rest of the line is treated as a comment and is ignored.
- Characters that appear in statements must be printable characters, unless otherwise noted. The character set is limited to the 26 alphabetic characters (uppercase and lowercase), the 10 numeric digits, and the following 18 special characters:

  plus (+)
  asterisk (*)
  slash (/)
  comma (,)
  period (.)
  ampersand (&)
  left and right parentheses [( )]
  straight single quote (')
  hyphen (-)
  equal (=)
  colon (:)
  straight double quote (")
  percent (%)

less than (<)
greater than (>)
question mark (?)
semicolon (;)

The following situations are exceptions to these rules:
- The MailAdministrator statement does not restrict any special characters
- The ExtWrtName statement allows only the following special characters:
  - dollar sign ($)
  - number sign (#)
  - at sign (@)

- Statements that are allowed only once must be specified only once. When a single statement is repeated, a warning message is written to the log file, and the last instance of the statement is used.
- Parameters that are allowed only once must be specified only once. If a single parameter is repeated, then the last instance of the parameter value is used.
- Specify multiple type statements and attributes based on the maximum allowed. When a statement or attribute is repeated more than the maximum number allowed, a warning message is written to the log file. The first instances, up to the maximum number of instances that are allowed, are used.
- Statements that contain braces ({ }) must specify the braces on separate lines. For example:

```
TargetServer
{
   TargetIp   9.66.103.222
}
```

- Any IP address reference can be either an IPv4 format or IPv6 format IP address when the stack is running in IPv6-enabled mode.
- At least one valid IP address or target name is required for the configuration to be valid. See "TargetServer statement" on page 1384 for details about configuration.
- Any warning that is detected during parsing causes messages to be written to the log and a single warning message to be written to the console. The new configuration is installed.
- You can use static system symbols in CSSMTP configuration file statements.

Results:
- If a configuration error is detected during startup, then CSSMTP writes an error message to the log and console, and exits.
- If a configuration error is detected during a dynamic refresh, then the entire refresh is rejected, an error message is written to the log and console, and CSSMTP continues running with the old configuration values.
- CSSMTP terminates in the following situations:
  - Start option errors are detected during initialization
  - Configuration file does not exist at initialization
  - Configuration file errors are detected during initialization
  - JES is not available during initialization
  - JES becomes unavailable while CSSMTP is processing the mail messages
  - The stop command is issued
  - The mail directory for extended retry becomes unusable

Starting CSSMTP

Use the S CSSMTP command on an MVS console or System Display and Search Facility (SDSF) to start CSSMTP.

Rules:

- You must start CSSMTP from a started procedure. A sample started procedure is included in member CSSMTP in SEZAINST. A configuration file is required. A sample CSSMTP configuration file is included in member CSSMTPCF in SEZAINST.
- Multiple instances of CSSMTP with different job names can be started with or without stack affinity. Each instance of CSSMTP that is running must be configured with a different external writer name. See "ExtWrtName statement" on page 1375 for more information.

The following options (in the started procedure) apply to CSSMTP:

-p | -P *tcpipJobName*

The *tcpipJobName* parameter is used in a common INET configuration to choose a socket stack for CSSMTP. It is also used for resolver functions. The environment variable _BPXK_SETIBMOPT_TRANSPORT can also specify the *tcpipJobName*. The -p start option overrides the environment variable. If neither form is used to set the *tcpipJobName*, then no affinity is used in a common INET configuration.

Results: In a Common INET configuration, there might be more than one TCP/IP stack. CSSMTP acts as a TCP/IP client. In this type of environment, you might want to associate CSSMTP with a specific stack, especially when there are multiple instances of CSSMTP to be started.

- The following list shows the priority for establishing TCPIP affinity for socket functions:
  1. Start option **-p** (CSSMTP sets affinity before any socket or resolver calls are made)
  2. Environment variable _BPXK_SETIBMOPT_TRANSPORT
  3. No affinity
- The following list shows the priority for establishing TCP/IP affinity for resolver functions:
  1. Application affinity.
  2. _BPXK_SETIBMOPT_TRANSPORT.

For resolver affinity information, see z/OS Communications Server: IP Configuration Guide.

-f |-F

This start option indicates that CSSMTP performs a cold start and flushes any checkpoint records from the previous execution of CSSMTP. To use checkpointing, you must set the CHKPOINT DD statement in the started procedure. The default is to use the checkpoint records to restart JES spool files at their last known status.

Tip: Member CSSMTPVL in SEZAINST is a sample job that you can use to allocate a VSAM linear data set that CSSMTP can use for checkpointing.

Result: If a valid CHKPOINT DD statement is not configured, checkpointing is not performed.

For resolver affinity, see z/OS Communications Server: IP Configuration Guide.

If CSSMTP cannot successfully parse the start options, log output is written to stdout and the application exits.

If you want time values that are generated by CSSMTP to appear in local time, you must set the TZ environment variable. If you do not set the TZ environment variable, timestamps created by CSSMTP are in Universal Time Coordinated (UTC) by default.

Set the TZ and configuration code page environment variables in one of the following ways:

- Specify TZ using the ENVAR parameter on the PARM statement in the started procedure. For example, use the following code:
  ```
  // PARM=('POSIX(ON) ALL31(ON)',
  // 'ENVAR("CSSMTP_CODEPAGE_CONFIG=IBM-1047"',
  // '"TZ=EST5EDT")/')
  ```
- Export the TZ and configuration code page environment variables in a file specified with the STDENV DD statement. For example, use the following code:
  ```
  //STDENV DD PATH='/etc/cssmtp.env',PATHOPTS=(ORDONLY)
  ```
  In the /etc/cssmtp.env file, use the following code:
  ```
  TZ=EST5EDT
  CSSMTP_CODEPAGE_CONFIG=IBM-1047
  ```

See z/OS Language Environment Programming Guide for more information about specifying runtime options and environment variables. See z/OS UNIX System Services Command Reference for details about setting the TZ environment variables.

CSSMTP sample started procedure

This topic contains a copy of the sample procedure.

```
//CSSMTP   JOB   JESLOG=(SPIN,'00:00'),MSGCLASS=A
//*
//*  JESLOG=(SPIN,'00:00')
//*     Spin the jeslog files once a day at midnight.
//*     This closes out the current JES joblog and creates a new one.
//*
//*  Use 'opt' to pass in parameters.  Example:   s cssmtp,opt='-f'
//CSSMTP   PROC  OPT=''
//*
//*   IBM Communications Server for z/OS
//*   SMP/E distribution name: EZAMLSAM
//*
//*   Licensed Materials - Property of IBM
//*   5650-ZOS
//*   Copyright IBM Corp. 2009, 2013
//*   Status = CSV2R1
//*
//*   Function: Sample procedure for running the
//*             CSSMTP application
//*
//*   This example shows no input parameter, but -p, -f or both
//*   can be used here
//*
//CSSMTP   EXEC PGM=CSSMTP,REGION=0K,TIME=NOLIMIT,
//      PARM=('ENVAR("_CEE_ENVFILE_S=DD:STDENV")/&OPT')
//*
//* Environment variables:
```

```
//* - Provide environment variables to run with the desired
//*   configuration.  As an example, the data set or file specified by
//*   STDENV could contain:
//*
//*   TZ=EST5EDT
//*   CSSMTP_CODEPAGE_CONFIG=IBM-1047
//*
//*   If you want to include comments in the data set or
//*   z/OS UNIX file, specify the _CEE_ENVFILE_COMMENT
//*   environment variable as the first environment variable
//*   in the data set or file.  The value specified for
//*   the _CEE_ENVFILE_COMMENT variable is the comment character.
//*   For example, if you want to use the pound sign, #, as
//*   the comment character, specify this as the first
//*   statement:
//*     _CEE_ENVFILE_COMMENT=#
//*
//STDENV   DD PATH='/etc/cssmtp.env',PATHOPTS=(ORDONLY)
//*STDENV DD DSN=TCPIP.TCPPARMS(CSSMTPEV),DISP=SHR
//*
//*
//* - The CSSMTP requires a configuration file.  If
//*   DD statement not configured, then the default is
//*   <jobname>.CSSMTP.CONF.   The configuration file can be a
//*   member of an MVS PDS(E), an MVS sequential file,
//*   or a z/OS UNIX file.
//*   See tcpip.SEZAINST(EZAMLCNF) for a sample configuration
//*
//CONFIG DD DSN=TCPIP.TCPPARMS(CSSMTP),DISP=SHR
//*CONFIG DD DSN=TCPIP.CONFIG.CSSMTP,DISP=SHR
//*CONFIG DD PATH='/etc/cssmtp.conf',PATHOPTS=(ORDONLY)
//*
//* - Output written to stdout and stderr goes to the data set or
//*   file specified with or SYSOUT, respectively.
//*   Normally, CSSMTP doesn't write output to stdout or stderr, but
//*   instead, output is written to the log file, which is specified
//*   by LOGFILE DD statement, and defaults to syslog
//*   daemon.  Severe startup errors, such as incorrect options
//*   specified, or being unable to open the log file, log output
//*   is instead written to stdout.
//*
//* - The logfile file can be a MVS sequential file, a z/OS UNIX file,
//*   SYSOUT or syslog daemon.  The default is syslog daemon.
//*
//* - If multiple CSSMTP are logging to the same log file, the user
//*   should use different log files or SYSLOGD.   Using the same
//*   log file from different applications can produce unpredictable
//*   results.
//*
//LOGFILE   DD PATH='/tmp/cssmtp.log',
//             PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*LOGFILE  DD SYSOUT=*,
//*            DCB=(RECFM=VB,LRECL=1028,BLKSIZE=3120)
//*LOGFILE  DD DSN=USER140.CSSMTP.LOG2,DISP=(MOD,CATLG),
//*            SPACE=(TRK,(10)),DCB=(RECFM=VB,LRECL=1028,BLKSIZE=3120)
//*
//* - Input/Output VSAM linear file that contains JES
//*   checkpoint information.   This is used if CSSMTP is not started
//*   with -f.
//*
//*   The Chkpoint file is a VSAM linear file.   If not configured
//*   then no checkpointing is done for JES spool files.
//*   See the sample JCL in CSSMTPVL to allocate the checkpoint
//*   data set.
//*
//*CHKPOINT DD DSN=TCPIP.CSSMTP.CHKPOINT,DISP=SHR
```

```
//*
//* - Output written to stdout and stderr goes to the data set or
//*   file specified with SYSOUT, respectively.
//SYSOUT DD SYSOUT=*
//*
//* - SYSTCPD explicitly identifies which data set is to be
//*   used to obtain the parameters defined by TCPIP.DATA
//*   when no GLOBALTCPIPDATA statement is configured.
//*   See the IP Configuration Guide for information on
//*   the TCPIP.DATA search order.
//*   The data set can be any sequential data set or a member of
//*   a partitioned data set (PDS).
//*
//*SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
//* - SYSTCPT is used to receive a detailed trace on how Resolver
//*   is resolving target servers.
//*SYSTCPT  DD SYSOUT=*
//          PEND
//CSSMTP    EXEC CSSMTP
```

Figure 56. CSSMTP application sample start procedure

CSSMTP configuration statements

Table 103 lists CSSMTP configuration file statements.

Table 103. CSSMTP configuration statements

Configuration file statement	Default	Required or optional	Update allowed by modify refresh	Purpose
BadSpoolDisp	Hold	Optional	Yes	Specifies the action to be taken when errors are encountered while the JES spool file is being processed.
ChkPointSizeLimit	64000	Optional	No	Specifies the number of concurrent mail messages for which checkpoint information is saved.
ExtendedRetry	• Age 5 • Interval 30 • MailDirectory /var/cssmtp/*extwrtname*/mail/	Optional	Yes (except MailDirectory)	Specifies the limits that CSSMTP uses when it attempts to resend mail messages that are not immediately deliverable after RetryLimit processing.
ExtWrtName	task job name	Optional	No	Specifies the external writer name that is used by CSSMTP for selection criteria when interfacing with the JES2 or JES3 subsystems.

Table 103. CSSMTP configuration statements (continued)

Configuration file statement	Default	Required or optional	Update allowed by modify refresh	Purpose
Header	• Date Yes • UserInfo Yes	Optional	Yes	Specifies the action to be taken when creating RFC 2822 mail headers.
JESJobSize	0 (unlimited)	Optional	Yes	Specifies the maximum data set size that is accepted from the JES spool file in thousands of bytes.
JESMsgSize	0 (unlimited)	Optional	Yes	Specifies the maximum mail message size that is accepted from a JES spool file, in thousands of bytes.
JESSyntaxErrLimit	5	Optional	Yes	Specifies the maximum number of syntax errors that are acceptable in a JES spool file before the rest of the JES spool file processing is stopped.
LogLevel	7	Optional	Yes	Specifies the level of logging and tracing.
MailAdministrator	No e-mail address is configured to send a report.	Optional	Yes	Specifies an e-mail address to which CSSMTP delivers reports for certain errors. This statement can be specified up to four times in a configuration file to deliver reports to multiple administrators.
Options	NullTrnc No	Optional	Yes	CSSMTP options
Report	Sysout	Optional	Yes	Specifies the action to be taken when problems are reported with JES spool files.
RetryLimit	Interval 1 Count 5	Optional	Yes	Specifies the limits that CSSMTP uses when attempting to re-send mail messages that are not immediately deliverable.
SMF119	No SMF recording	Optional	Yes	Specifies the records to be written to SMF.

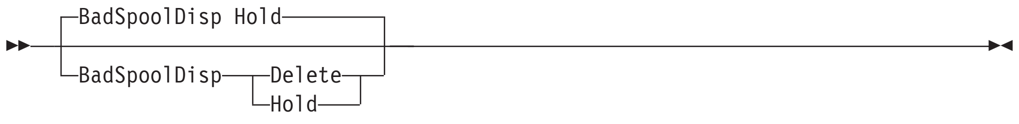
Table 103. CSSMTP configuration statements (continued)

Configuration file statement	Default	Required or optional	Update allowed by modify refresh	Purpose
TargetServer	• ConnectPort 25 • ConnectLimit 5 • MaxMsgSent 0 • MessageSize 524288 • Secure No You must provide a value for TargetIP, TargetName, or TargetMx.	Required	Yes	Specify one or multiple TargetServer statements to define target servers (resolved or configured IP addresses) and their connection attributes to which CSSMTP connects for sending mail.
Timeout	• AnyCmd 300 • ConnectRetry 120 • DataBlock 180 • DATACmd 120 • DataTerm 600 • InitialMsg 300 • MAILCmd 300 • RCPTCmd 300	Optional	Yes	Specifies the timeout values, in seconds, for the interaction between CSSMTP and a target server.
Translate	IBM-1047	Optional	No	Specifies the translation code page of the records read from the JES spool data set.
Undeliverable	ReturnToMailFrom Yes DeadLetterAction Store DeadLetterDirectory /var/cssmtp/ *extwrtname*/ deadletter/	Optional	Yes	Specifies the method to use for handling undeliverable mail.
UserExit	None	Optional	Yes	Controls whether this CSSMTP calls CSSMTP exit program provided by the customer to examine data being sent to CSSMTP from the JES spool data set.

BadSpoolDisp statement

Use the BadSpoolDisp statement to indicate to CSSMTP what to do with JES spool files when errors were encountered when processing the spool file. See the information about common terms in z/OS Communications Server: IP Configuration Guide for a description of bad spool file.

Syntax



Parameters

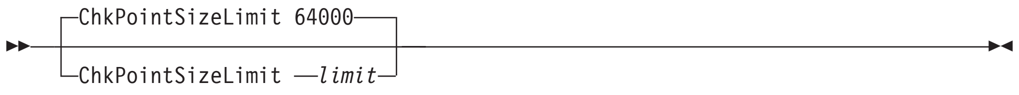
Delete
: Specify that CSSMTP should delete the spool file.

Hold
: Specify that CSSMTP should change the disposition of the spool file to HOLD so that CSSMTP cannot process it.

ChkPointSizeLimit statement

Use the ChkPointSizeLimit statement to specify the number of concurrent mail messages for which checkpoint information is saved. This saved information is used for a warm start so that CSSMTP does not reprocess the entire spool file during a restart. Checkpointing warm functions only when you are restarting CSSMTP with the same job name and external writer name.

Syntax



Parameters

limit
: An integer value in the range 64 000 - 512 000 that represents the number of concurrent mail messages that can have checkpoint information saved. The default size is 64 000.

  Tip: If the CHKPOINT data set is not allocated, this value is ignored.

  Result: If an update to the ChkPointSizeLimit statement is detected during a dynamic refresh, CSSMTP continues to run using the previous ChkPointSizeLimit value and a warning message is written to the log and console.

ExtendedRetry statement

Use the ExtendedRetry statement to extend the retry processing that Communications Server SMTP (CSSMTP) uses when it attempts to resend mail messages that are not immediately deliverable. For basic information about extended retry, see the information about common terms in z/OS Communications Server: IP Configuration Guide.

The RetryLimit statement defines the retry interval during which CSSMTP will try sending the mail messages again while the JES spool file is still available. When the long retry interval expires (the interval is the product of multiplying the COUNT and INTERVAL values), the mail messages are copied to a z/OS UNIX directory for extended retry processing and the JES spool file is released. When the extended retry interval expires, the mail messages become undeliverable.

After both the long retry interval and the extended retry interval retries have expired, the following actions occur when information about the mail sender's address is available:

1. CSSMTP uses the setting on the ReturnToMailFrom parameter in the UNDELIVERABLE statement to determine its next action.
2. CSSMTP attempts to send the undeliverable mail notifications to the originator of the mail message through the configured target servers. If CSSMTP cannot send the notification on the first try, the mail message becomes a dead letter (see “UNDELIVERABLE statement” on page 1389) and no retries will be made.

Guideline: If the ExtendedRetry statement is not defined in the configuration file, extended retry processing is inactive.

Tip: You can set the Interval parameter on the ExtendedRetry statement to 0 to bypass extended retry processing.

Syntax

```
►►──ExtendedRetry──┤ Put Braces and Parameters on Separate Lines ├──────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{───────────────────────────┬────────────────────────────────────────────┤
   ├─┤ ExtendedRetry Parameters ├─┤
   └─}───────────────────────────┘
```

ExtendedRetry Parameters:

```
   ┌─Age 5────┐  ┌─Interval 30──────┐
├──┼──────────┼──┼──────────────────┼──────────────────────────────────────────►
   └─Age days─┘  └─Interval minutes─┘

   ┌─MailDirectory /var/CSSMTP/extwrtname/mail/─┐
►──┼─────────────────────────────────────────────┼────────────────────────────┤
   └─MailDirectory mailDirectory─────────────────┘
```

Parameters

Age *days*
: Indicates the number of days that CSSMTP attempts to resend mail messages after the extended term retry interval expires. The default value is 5 days. The value 0 means that an unlimited number of retry attempts will be made. The maximum value is 2147483647.

Interval *minutes*
: Indicates the length of time, in minutes, that CSSMTP waits between subsequent attempts to resend mail messages. The default value is 30 minutes. The maximum value is 1440 minutes (one day).

  If the Interval value is 0, no extended retry will be performed and a mail message becomes undeliverable if CSSMTP cannot deliver it to the target servers on the first try.

MailDirectory *mailDirectory*

The fully qualified directory name in the z/OS UNIX file system where CSSMTP stores the mail message when the Interval parameter is set to a nonzero value. Valid values are 1 - 512 characters in length and must begin with a slash (/) to define the fully qualified directory. The default mail directory is `/var/cssmtp/`*`extWrtName`*`/mail/`, where *extWrtName* is the name specified in the ExtWrtName statement.

Results:

- A slash is added to the end of the mail directory name if it is not present.
- If the mail directory does not exist it is created.

Guidelines:

- The user ID that CSSMTP starts under must have read/write access to this directory. If you are running multiple CSSMTP servers with extended retry enabled, each server should have its own directory and should not have access to the directories of other servers.
- Ensure that there is adequate space in the filesystem for extending retry processing. For example, if each mail message is an average of 8000 bytes in size and you expect to have 1000 mail messages to be tried again, then about 16 million bytes of file space is needed [2 (files per mail message) × 8K (z/FS file allocation size) × 1000]. Factor in the 75% threshold for message EZD1862I, then the file system should be about 22 million bytes in size (plus a margin of error).
- The size of the MailDirectory is the sum of all of the size of all the mail messages saved for extended retry at any one time.

Restrictions:

- If extended retry processing is activated, you must stop and restart CSSMTP in order to change the name of the mail directory.
- If CSSMTP is active, you cannot change the name of the mail directory and you cannot delete or change the files.
- The content of the files created by CSSMTP and stored in the mail directory cannot be modified.

Tip: You can configure your mail directory to be on a different z/OS UNIX file system than where you have configured your deadletter directory. By doing this, you can manage the resource better. For more information, see "UNDELIVERABLE statement" on page 1389.

ExtWrtName statement

Use the ExtWrtName statement to specify the external writer name used by Communications Server SMTP (CSSMTP) as selection criteria when CSSMTP is interfacing with the JES2 or JES3 subsystems.

Syntax

```
      ┌─ExtWrtName wrtName─┐
►►────┴────────────────────┴─────────────────────────────────────────────►◄
```

Parameters

wrtName

A string 1 - 8 characters in length that specifies the external writer name to be used by CSSMTP. This value can contain alphanumeric characters, as well as the special characters $ # @. The default value is the CSSMTP job name. The writer name is not case sensitive.

Results:

- If an ExtWrtName value is not configured, CSSMTP sets this parameter value to CSSMTP job name.
- If an update to the ExtWrtName statement is detected during a dynamic refresh, CSSMTP continues to run using the previous ExtWrtName value and a warning message is written to the log and console.

Tip: The SMTPD job name should not be specified for CSSMTP external writer name if the SMTPD gateway is running on the same LPAR. The results are unpredictable.

Restrictions:

- Do not code INTRDR, STDWTR, or NJERDR because these names are reserved for JES.
- If the JES2 DESTDEF statement specifies NODENAME=REQUIRED, then the writer name specified in the ExtWrtName statement must be defined to JES2. You can dynamically define it to JES using the following command:

  ```
  $ADD DESTID(xxxxxxxx),DEST=xxxxxxxx
  ```

  where xxxxxxxx is the *wtrName* specified in the ExtWrtName statement. See z/OS JES2 Initialization and Tuning Reference for information about the DESTDEF statement. To permanently add the destination use the JES initialization DEStid statement:

  ```
  DESTID(xxxxxxxx) DEST=xxxxxxxxx
  ```

  See z/OS JES2 Initialization and Tuning Reference for information about the DESTID JES2 statement. If JES2 DESTDEF specifies NODENAME=OPTIONAL then the writer name specified on ExtWtrName does not need to be specified.
- You cannot start multiple CSSMTP applications that use the same external writer name.

Header statement

Use the Header statement to change the behavior of CSSMTP when you create RFC 2822 mail headers. The mail headers are created when the JES spool file is processed. If the Header parameters are modified, mail headers that have been created are not altered.

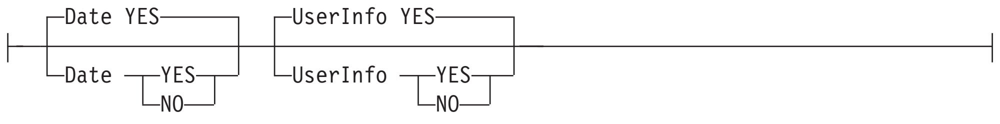
Syntax

```
►►──Header──┤ Put Braces and Parameters on Separate Lines ├──────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────┬──────────────────────────────────────────────┤
   ├─┤ Header Parameters ├─┤
   └─}─────────────────────┘
```

Header Parameters:



Parameters

`Date`
: Specifies whether CSSMTP adds the Date: header if the header is missing.

  NO The Date: header is not inserted into the mail message by CSSMTP.

  YES The Date: header is inserted into the mail message by CSSMTP. This is the default value.

`UserInfo`
: Specifies whether user information is included in Mail headers that CSSMTP creates.

  NO User information is not inserted into the mail message by CSSMTP. This setting applies to the following information:
  - If CSSMTP creates the Message-ID: header, the job name of the mail message of the JES spool file and the job identifier of the JES spool file are not included in the Message-ID: header.
  - The CSSMTP JobName is inserted in the for field of the Received: header instead of the JES origin USER ID and the optional Notify USER ID.
  - The job name of the mail message of the JES spool file and the job identifier of the JES spool file are not included in the Id field of the Received: header.

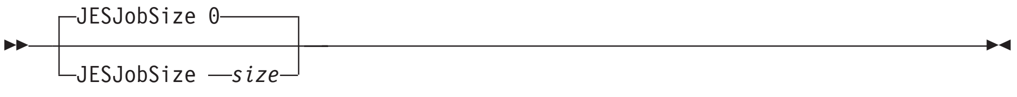
  YES User information is inserted into the mail message by CSSMTP. This is the default value.

JESJobSize statement

Use the JESJobSize statement to specify the maximum data set size that is accepted from the JES spool file, in thousands of bytes.

Tip: Set this value for the largest spool job expected and add enough room for future growth.

Syntax



Parameters

`size`
: An integer value in the range 0 - 1 000 000 that represent the maximum data set size that is accepted from the JES spool file, in thousands of bytes. The default size is 0, which specifies an unlimited data set size.

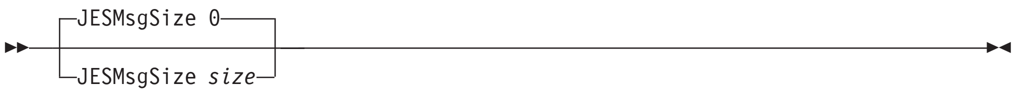
Restriction: The JESJobSize value must be greater or equal to the JESMsgSize value.

JESMsgSize statement

Use the JESMsgSize statement to specify the maximum mail message size accepted from a JES spool file, in thousands of bytes.

Tip: Set this value for the largest mail message expected and add enough room for future growth.

Syntax



Parameters

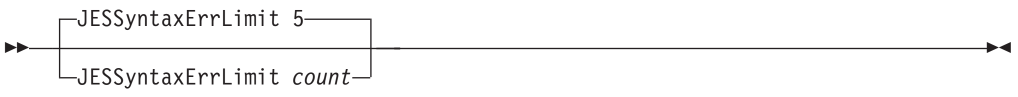
size
: An integer value in the range 0 - 1 000 000 that represents the maximum mail message size accepted from a JES spool file, in thousands of bytes. The default size is 0, which specifies an unlimited data set size.

  Restriction: The JESMsgSize value must be less than or equal to the JESJobSize value.

JESSyntaxErrLimit statement

Use the JESSyntaxErrLimit statement to specify the maximum number of syntax errors that are acceptable in a JES spool file.

Syntax



Parameters

count
: An integer in the range 0 - 999 that represents the maximum number of syntax errors that are acceptable in a JES spool file. The default value is 5. The value 0 indicates that an unlimited number of syntax errors is acceptable. If you set a high or unlimited value on this statement, then a waste of system resources such as CPU use can occur when the JES spool file contains many errors.

  Tip: Syntax errors in the RCPT commands are not counted against the JESSyntaxErrLimit statement. Instead, a syntax error in a RCPT command is treated as an undeliverable mail message, if the rest of the SMTP commands are valid. Use the “UNDELIVERABLE statement” on page 1389 to control the handling of undeliverable mail.

LogLevel statement

Use the LogLevel statement to specify the level of logging and tracing.

Syntax

```
     ┌─LogLevel 7─────┐
►►───┼────────────────┼───────────────────────────────────────────────────►◄
     └─LogLevel level─┘
```

Parameters

level

Specifies the log *level* . The *level* value represents a particular log level or combination of debug levels. Possible values are:

0	No messages are logged.
1	Error-level messages are logged.
2	Warning-level messages are logged.
4	Event-level messages are logged.
8	Information-level messages are logged.
16	JES-level messages are logged. This value traces Communications Server SMTP (CSSMTP) commands and command syntax parser replies between the JES spool file and CSSMTP.
32	Network-level messages are logged. This traces CSSMTP commands and remote SMTP server replies between CSSMTP and the TCP/IP network.
64	Debug-level messages are logged. These messages are internal debug messages intended for development and IBM service use only.
128	Trace-level messages are logged. These messages are function entry and exit traces that show the path through the code. This level is intended for development and IBM service use only.

Guideline: To log a combination of log levels, add the log level numbers and specify the resulting value. During the initialization phase of the application the log level is set to 127 to capture any initialization problems. When initialization is complete, then the log level value is set to what is coded on this statement. The default log level is 7, which captures all error, warning, and event messages after the initialization is complete.

MailAdministrator statement

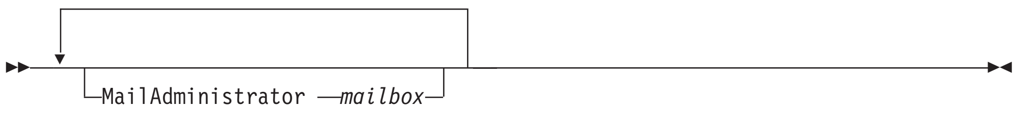
Use the MailAdministrator statement to specify an e-mail addresses with the format *userid@host.domain* (mailbox) to which Communications Server SMTP (CSSMTP) delivers error reports. See the information about common terms in z/OS Communications Server: IP Configuration Guide for a description of mail administrator. Error reports are generated by CSSMTP when a problem is detected while processing a spool file from the JES subsystem (see the “REPORT statement” on page 1381).

Results:

- Only the first four MailAdministrator statements are used. If more than four MailAdministrator statements are configured, CSSMTP issues a warning message to the console and log.

- When configured (see the “REPORT statement” on page 1381), a report is sent to each mail administrator for each spool file that contains errors. This report is in the form of one e-mail with multiple recipients.
- When a MODIFY REFRESH command is issued and the order of the MailAdministrator statements or any of the mail addresses has changed, the configuration is updated.

Syntax



Parameters

mailbox
: The mail administrator's e-mail address to which the CSSMTP delivers error reports. There is no default value

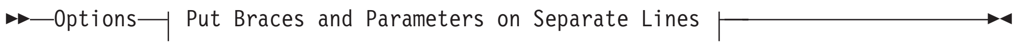
Restrictions:

- Duplicate mailbox names are not allowed.
- The *mailbox* value is case sensitive.
- The *mailbox* value must be defined as *userid@host.domain* for the mail address. The *userid* value can be 1 - 64 characters in length. The *host.domain* value can be 1 - 255 characters in length.

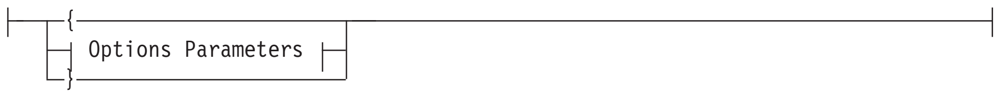
Options statement

Use the Options statement to change the processing behavior of CSSMTP.

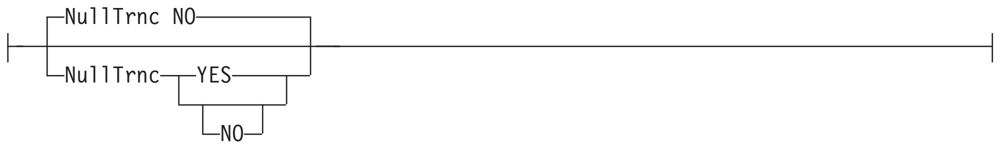
Syntax



Put Braces and Parameters on Separate Lines:



Options Parameters:



Parameters

NullTrnc
Specifies whether the trailing null characters are stripped from mail command records. The mail command records include EHLO, HELO, MAIL, RCPT, DATA, and STARTTLS.

NO The trailing null characters are not stripped from the mail command records by CSSMTP. This is the default value.

YSE
The trailing null characters are stripped from the mail command records by CSSMTP.

REPORT statement

Use the REPORT statement to indicate the action you want to take for reporting problems with JES spool files. These problems include the following ones:

- Errors accessing or reading the spool file
- SAF (RACF) violations
- User exit rejection of a mail message or the spool file
- Syntax errors in the spool file.
- Unsuccessful delivery. See"UNDELIVERABLE statement" on page 1389 for details.

Tip: For a description of an error report, see the example of an error report to MailAdministrator or the sysout file in z/OS Communications Server: IP Diagnosis Guide.

Syntax

```
     ┌─REPORT Sysout──────────┐
►►───┤                        ├──────────────────────────────────────────────►◄
     └─REPORT────┬─Admin──┬───┘
                 ├─None───┤
                 └─Sysout─┘
```

Parameters

Admin
Indicates that an error report should be sent to the configured mail administrators.

Tip: To avoid losing the error report information in the event that it cannot be delivered to one or more configured mail administrators, code DeadLetterAction store in the "UNDELIVERABLE statement" on page 1389.

Requirement: At least one mail administrator must be defined.

None
Indicates that no error reports should be created.

Without a report, the log must be inspected for messages about any problems found in the JES spool file.

Sysout
Indicates that CSSMTP should create a sysout file that contains the report.

RetryLimit statement

Use the RetryLimit statement to set the limits that Communications Server SMTP (CSSMTP) uses when attempting to resend mail messages that are not immediately deliverable. See the information about common terms in z/OS Communications Server: IP Configuration Guide for description.

After the retry limit specified on this statement is exhausted the following actions occur:

- CSSMTP uses the setting on the Interval parameter of the ExtendedRetry statement to determine whether the mail message is eligible for extended retry. If the mail message is eligible, CSSMTP sends it for extended retry processing. If the mail message is not eligible for extended retry, the message goes immediately to the undeliverable process as specified by the "UNDELIVERABLE statement" on page 1389.
- CSSMTP uses the setting in the ReturnToMailFrom parameter in the UNDELIVERABLE statement to determine its next action when there is information regarding the mail sender's address.
- CSSMTP attempts to send the undeliverable mail notifications to the originator of the mail message, through the configured target servers. If the notification cannot be sent on the first try, it becomes a dead letter (see "UNDELIVERABLE statement" on page 1389). No retries are made.

See "ExtendedRetry statement" on page 1373 for a description of the actions that you need to take when a mail message is eligible for extended retry processing.

Syntax

```
►►──RetryLimit──┤ Put Braces and Parameters on Separate Lines ├──────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬────────────────────────────────────────────┤
   ├─┤ RetryLimit Parameters ├─┤
   └─}──────────────────────────┘
```

RetryLimit Parameters:

```
   ┌─Count 5──────┐  ┌─Interval 1───────┐
├──┼──────────────┼──┼──────────────────┼────────────────────────────────────┤
   └─Count number─┘  └─Interval minutes─┘
```

Parameters

Count *number*
: Indicates the number of times CSSMTP attempts to resend mail. Valid values are in the range 0 - 120. The default is 5.

Interval *minutes*
: Indicates the amount of time, in minutes, that CSSMTP waits before attempting to resend mail. Valid values are in the range 0 - 120. The default value is 1.

If the Count or Interval value is zero, then no long retry is performed, and mail becomes undeliverable if it cannot be delivered to the target servers on the first try.

Restriction: The total configured time cannot exceed 5 days (7200 minutes in total). For example, 120 retries with 60 minutes per retry results in a maximum time of 5 days.

Tip: During this time, the spool file remains in use. A high configured time total can result in excess spool and storage usage. You can use the MODIFY FLUSHRETRY,TKID= command to force mail messages off the retry queue earlier than the configured time allowed in the RetryLimit statement. If you do, the values in the RetryLimit statement are no longer used for these mail messages. If the mail messages cannot be sent, then the mail messages become undeliverable messages.

SMF119 statement

Use the SMF119 statement to indicate which SMF records are written by Communications Server SMTP (CSSMTP).

Syntax

```
►►──SMF119──┤ Put Braces and Parameters on Separate Lines ├──────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{─────────────────────┬──────────────────────────────────────────────┤
   ├─┤ SMF119 Parameters ├─┤
   └─}─────────────────────┘
```

SMF119 Parameters:

```
   ┌─Config NO────────┐  ┌─Connect NO────────┐  ┌─Mail NO────────┐
├──┴─Config──┬─NO──┬──┴──┴─Connect──┬─NO──┬──┴──┴─Mail──┬─NO──┬─┴──────────►
             └─YES─┘                └─YES─┘             └─YES─┘

   ┌─Spool NO────────┐  ┌─Stats NO────────┐
►──┴─Spool──┬─NO──┬──┴──┴─Stats──┬─NO──┬──┴──────────────────────────────┤
            └─YES─┘              └─YES─┘
```

Parameters

Config
Specifies if the configuration SMF records are written

NO The configuration SMF records are not written. This is the default.

YES
The configuration SMF records are written. The configuration SMF records are written at initialization and when the configuration is updated with the MODIFY REFRESH command.

Connect
Specifies if the connection SMF records are written

NO The connection SMF records are not written. This is the default.

YES
The connection SMF records are written. The connection SMF records are written at the end of each client connection used for mail data transfer to a target server.

Mail
Specifies if the mail message SMF records are written

NO The mail message SMF records are not written. This is the default.

YES
The mail message SMF records are written. The mail message SMF records are written at the completion of each mail message from the spool data set.

Spool
Specifies if the spool SMF records are written

NO The spool SMF records are not written. This is the default.

YES
The spool SMF records are written. The spool SMF records are written when all the mail messages for a JES spool file have been processed.

Stats
Specifies if the statistical SMF records are written. The statistical SMF records are written at the MVS SMF intervals and at CSSMTP termination. For information about SMF interval processing, see the section about INTVAL and SYNCVAL - Performing interval accounting in z/OS MVS System Management Facilities (SMF).

NO The statistical SMF records are not written. This is the default.

YES
The statistical SMF records are written.

TargetServer statement

Use the TargetServer statement to specify one or more target servers (resolved or configured IP addresses) and their connection attributes. CSSMTP establishes connections to the target servers in order to send mail.

Rules:
- If you are configuring the TargetIP parameter, the TargetName parameter, or both, then you can configure multiple TargetServer statements.
- When you issue a MODIFY REFRESH command, if the order of the target servers changes, the configuration is updated.
- Each TargetIP or TargetName must be unique.

Results:
- Only the first four unique TargetIP values, TargetName values, or both values are used. If more than four values are configured, the application issues a console message and logs a warning. The four target servers that are selected are based on the configuration order in which the parameters were configured. CSSMTP only uses the first four IP addresses.
- If duplicate target server IP addresses are resolved from the configured TargetIP or from the TargetName IP addresses, CSSMTP issues a console message and logs a warning.

- If the TCP/IP stack supports only IPv4, any configured IPv6 addresses are ignored and the application issues a console message and logs a warning.

Restrictions:

- You must define at least one TargetServer statement, and it must contain at least one TargetIP, TargetName, or TargetMx parameter.
- Only four TargetServer statements or the first four TargetIP and TargetName parameters with TargetServer statements are used.
- If distinct target servers can be reached by way of a single IP address, the target servers must have the same capabilities. For example, if a dynamic VIPA (DVIPA) address is specified, the mail servers for that DVIPA address must have the same capabilities. In this example, all the servers must be ESMTPs or SMTPs, but not both, that have the same capabilities.
- You can configure only one TargetMx parameter.

Syntax

```
►►──TargetServer──┤ Put Braces and Parameters on Separate Lines ├──────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├──┬─{──────────────────────────┬─────────────────────────────────────────────┤
   ├─┤ TargetServer Parameters ├┤
   └─}──────────────────────────┘
```

TargetServer Parameters:

```
      ┌───────────────────┐                             ┌─ConnectPort 25───┐
      ▼                   │                             │                  │
├──────┬─────────────────┬┴───┬───────────────────┬─────┼──────────────────┼──────►
       ├─TargetIP addr───┤    └─TargetMx mxName───┘     └─ConnectPort port─┘
       └─TargetName name─┘

   ┌─ConnectLimit 5─────┐  ┌─MaxMsgSent 0──────┐  ┌─MessageSize 524288─┐
►──┼────────────────────┼──┼───────────────────┼──┼────────────────────┼──────────►
   └─ConnectLimit count─┘  └─MaxMsgSent count──┘  └─MessageSize size───┘

   ┌─Secure NO──────┐
►──┼────────────────┼────────────────────────────────────────────────────────────┤
   └─Secure──┬─NO──┬┘
             └─YES─┘
```

Parameters

ConnectPort
: Defines the port that CSSMTP uses to connect to a target server.

  Result: If an update to the port is detected during a dynamic refresh, then CSSMTP must terminate all the active connections to the target servers that are associated with this TargetServer statement in order to use the new port value.

  Requirement: This port must match the listening port number used by the target server.

The valid range of port values is 1 - 65535. The default port is 25.

ConnectLimit
Limits the number of concurrent socket connections to the target server from CSSMTP. This might be useful if your server has a concurrent connection limit. One of the socket connections is used by CSSMTP to monitor the SMTP server.

The valid values are in the range 2 - 5. The default limit is 5 connections.

Result: If an update to the ConnectLimit value is detected during a dynamic refresh, CSSMTP must terminate all the active connections to the target servers that are associated with this TargetServer statement in order to use the new limit.

MaxMsgSent
Specifies the maximum number of mail messages that can be sent on a single connection. When the MaxMsgSent value is exceeded, CSSMTP closes the connection to the target server and reestablishes a new connection to the same target server. Mail messages continue to be sent over the new connection until the MaxMsgSent value is reached again.

The valid range is 0 - 2147483647. The default is 0, which means that an unlimited number of messages can be sent.

MessageSize
Specifies the maximum size of a mail message that can be sent to target servers that do not support ESMTP size extensions.

Valid values are in the range 1000 - 2147483647. The default size is 524288 bytes (512 KB).

Secure
Indicates whether Transport Layer Security (TLS) is required between the client and a target server. TLS provides private, authenticated communication over the internet. See the steps for setting up security for CSSMTP in z/OS Communications Server: IP Configuration Guide.

Result: If an update to the secure value is detected during a dynamic refresh, CSSMTP must terminate all the active connections to the target servers that are associated with this TargetServer statement in order to use the new secure value.

NO TLS is not required between CSSMTP and a target server. However, if the STARTTLS SMTP command is used in the spool file, a TLS connection is attempted with this server.

YES
TLS is required between CSSMTP application and a target server. If a TLS session cannot be established, then the server is not usable.

TargetIP
The IPv4 or IPv6 address of the target server to which CSSMTP connects.

Restrictions:

- IPv4-mapped IPv6 addresses and IPv6 addresses with the reserved prefix ::/96 are not valid.
- The IPv6 unspecified address (::0) and IPv4 unspecified address (0.0.0.0) are not allowed.
- Duplicate IP addresses are not allowed.

TargetName
The host name or fully qualified host name used for a resolver A or AAAA

query. Valid values are 1 - 255 characters in length. If the host name is used, the resolver appends the domain information, which is obtained from the TCPIP.DATA data set.

The TargetName value is not case sensitive.

Result: CSSMTP only uses the first four configured or resolved IP addresses to send the mail message. If more than four target servers are found, this application issues a console message and logs a warning.

Restriction: Duplicate host names are not allowed.

TargetMx

The name or a fully-qualified domain name used for a resolver MX query. This name might resolve into multiple MX records that include a preference value. The lower the preference value is, the more likely that the record is used. Valid values are 1 - 255 characters in length.

The lower the value, the higher the preference.

The TargetMx value is not case sensitive.

Results:

- Only the first configured TargetMx value is used. If more than one value is configured, the application issues a console message and logs a warning.
- Only the first four target servers are saved for the configured TargetMx value. If more than four target servers are returned, CSSMTP issues a console message and logs a warning.
- The lower preference is honored by not sending mail messages to those IP addresses unless the higher preference target servers are unavailable.

Restriction: The TargetMx parameter is mutually exclusive with the TargetName parameter and the TargetIP parameter. If you want to use this statement, MX records for this TargetMx value must exist in the DNS database. If no MX records are found then this is handled as an error and console message EZD1815E is generated.

TIMEOUT statement

Use the Timeout statement to define a set of timeout values, in seconds, for the interaction between CSSMTP and the target server.

Syntax

```
►►──Timeout──┤ Put Braces and Parameters on Separate Lines ├──────────────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├───┬─{───────────────────────┬──────────────────────────────────────────────────┤
    ├─┤ Timeout Parameters ├──┤
    └─}───────────────────────┘
```

Timeout Parameters:

```
   ┌─AnyCmd 300──────┐  ┌─ConnectRetry 120──────┐  ┌─DataBlock 180──────┐
├──┴─AnyCmd seconds──┴──┴─ConnectRetry seconds──┴──┴─DataBlock seconds──┴──────►
```

```
   ┌─DATACmd 120──────┐  ┌─DataTerm 600──────┐  ┌─InitialMsg 300──────┐
►──┴─DATACmd seconds──┴──┴─DataTerm seconds──┴──┴─InitialMsg seconds──┴──────────►

   ┌─MAILCmd 300──────┐  ┌─RCPTCmd 300──────┐
►──┴─MAILCmd seconds──┴──┴─RCPTCmd seconds──┴────────────────────────────────────┤
```

Parameters

AnyCmd
The length of time, in seconds, that CSSMTP waits for a response on any other SMTP command (for example, EHLO, HELO, RSET, QUIT, and STARTTLS) from the SMTP server.

Valid values are in the range 30 - 1 200. The default value is 300.

ConnectRetry
The length of time, in seconds, that CSSMTP waits before trying again to connect to a target server after a failed attempt.

Valid values are in the range 30 - 1 200. The default value is 120.

DataBlock
The length of time, in seconds, that CSSMTP waits for the TCP send call to complete while transferring a block of data to the TCP/IP stack .

Valid values are in the range 30 - 1 200. The default value is 180.

DATACmd
The length of time, in seconds, that CSSMTP waits for a response to the DATA command from the SMTP server.

Valid values are in the range 30 - 1 200. The default value is 120.

DataTerm
The length of time, in seconds, that CSSMTP waits for a response to the final period that terminates the mail message data from the SMTP server.

Valid values are in the range 30 - 1 200. The default value is 600.

InitialMsg
The length of time, in seconds, that CSSMTP waits for an initial response after the connection is established with the SMTP server.

Valid values are in the range 30 - 1 200. The default value is 300.

MAILCmd
The length of time, in seconds, that CSSMTP waits for a response to the MAIL command from the SMTP server.

Valid values are in the range 30 - 1 200. The default value is 300.

RCPTCmd
The length of time, in seconds, that CSSMTP waits for a response to the RCPT command from the SMTP server.

Valid values are in the range 30 - 1 200. The default value is 300.

TRANSLATE statement

Use the TRANSLATE statement to define the supported code page to be used to translate data received from the JES spool data set to ASCII for sending mail messages. See the code set conversions supplied in *z/OS Support for Unicode: Using Unicode Services* for details about the supported code pages.

Tip: The following POSIX variant characters are converted to ASCII using *iconv* determined by the configured EBCDIC code page. The POSIX variant characters are:

\ [ ] { } ^ ~ ! # | $ @ `

Syntax

```
      ┌─Translate IBM-1047─────────────┐
►►────┤                                ├──────────────────────────────────────►◄
      └─Translate code_page_identifier─┘
```

Parameters

`Translate`

Specifies the code page to be used for translating the spool file records. If you do not specify this statement, the default code page is IBM-1047. See Table 104 on page 1392 for a list of supported code pages.

Results:

- If an update to the Translate statement is detected during a dynamic refresh, CSSMTP continues to use the old translate value and a warning message is written to the log and the console.

Restrictions: The code page must have the following characteristics:

- Defined to Unicode System Services
- An EBCDIC code page
- A single byte code page (not double byte nor multibyte)
- A translation to and from IBM-1047
- A translation to and from ISO-8859-1
- Translations must be reversible
- The environment variable _ICONV_TECHNIQUE should be left undefined or have the value of its default LMREC.

See Table 104 on page 1392 for a list of code page identifiers.

UNDELIVERABLE statement

Use the UNDELIVERABLE statement to indicate to CSSMTP the method used for handling undeliverable mail. See the information about common terms in z/OS Communications Server: IP Configuration Guide more details.

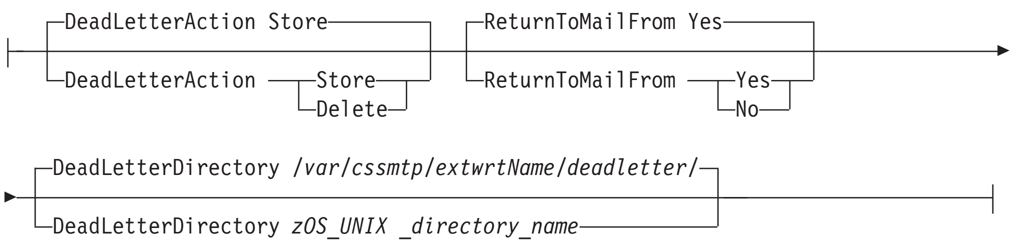
Syntax

```
►►──UNDELIVERABLE──┤ Put Braces and Parameters on Separate Lines ├──────────►◄
```

Put Braces and Parameters on Separate Lines:

```
├───┬─{────────────────────────────┬──────────────────────────────────────┤
    ├─┤ UNDELIVERABLE Parameters ├─┤
    └─}────────────────────────────┘
```

UNDELIVERABLE Parameters:



DeadLetterAction Store
DeadLetterAction
Store
Delete
ReturnToMailFrom Yes
ReturnToMailFrom
Yes
No
DeadLetterDirectory /var/cssmtp/extwrtName/deadletter/
DeadLetterDirectory zOS_UNIX _directory_name

Parameters

DeadLetterAction

Indicates to CSSMTP the action to take when a dead letter is detected. A dead letter is an undeliverable mail notification that cannot be returned to the original sender or that an error report, if requested, could not be delivered to the mail administrators. This can occur if the MAIL FROM: value is null in the original spool file and it cannot be sent. See the information about common terms in z/OS Communications Server: IP Configuration Guide for a description of dead letter.

Delete

Indicates that CSSMTP should not save the dead letter to a z/OS UNIX file system. If this option is chosen then it is recommended that you use the report statement. See the Sysout parameter in "REPORT statement" on page 1381 for information about recording original spool problems.

Store

Indicates that CSSMTP should store the mail message to the directory defined in the DeadLetterDirectory parameter. Each dead letter is stored as a separate file within the dead letter directory.

DeadLetterDirectory

The z/OS UNIX file system fully qualified directory name where CSSMTP creates the dead letter mail message that is stored when the DeadLetterAction parameter is set to Store. Valid values are 1 - 512 characters in length and must begin with a slash (/) to define the fully qualified directory. The default dead letter directory is /var/cssmtp/*extWrtName*/deadletter, where the *extWrtName* is the name used for ExtWrtName. See the information about common terms in z/OS Communications Server: IP Configuration Guide for a description of a dead letter.

Result: An ending slash (/) is added to the directory name if not configured.

ReturnToMailFrom

Indicates whether CSSMTP should create an undeliverable mail notification to be returned to the originator. The undeliverable mail notification contains the original mail message as well as additional information indicating the reason for the failure. This parameter does not apply to original mail messages that do not have the originator specified on the MAIL FROM SMTP command (for example, MAIL FROM:<>). In this case, if the original mail message cannot be sent, it immediately becomes a dead letter.

YES

CSSMTP creates the undeliverable mail notification to be returned to the originator with additional information regarding the reason for the failure.

Result: If the original spool file contains no errors other than undeliverable errors, the spool file is deleted.

NO CSSMTP does not create the undeliverable mail notification to be returned to the originator. If you specify this parameter, you should use the REPORT statement with a value of *sysout* or *admin*.. See "REPORT statement" on page 1381 for information about how to create a report that the original mail message is undeliverable.

The action taken on the original spool file that contained undeliverable errors is based on the value configured on the BadSpoolDisp statement. See "BadSpoolDisp statement" on page 1372 for details.

For a description of the undeliverable status, see the information about mail problems in z/OS Communications Server: IP Configuration Guide.

Results:

- If DeadLetterAction is set to Store:
  - If the DeadLetterDirectory parameter, to set the dead letter directory, is set to /userDirectory/deadLetterDirectory, then CSSMTP creates the userDirectory, deadLetterDirectory, or both in the z/OS UNIX file system if they do not already exist. The following name is the file name that would be used for the dead letter that is stored on the configured directory. The file name, TESTMAIL.SYS00006.Sep302008.160454.541437.1U, is constructed from the message ID of the mail message in the original spool file with the letter *U* appended and the fully-qualified host name removed. See the SMTP command and data command information in z/OS Communications Server: IP User's Guide and Commands for details about the Message-ID.

    The following sample shows a dead letter directory (/userDirectory/deadLetterDirectory) that contains two dead letters:

```
/userDirectory/deadLetterDirectory/TESTMAIL.SYS00006.Sep302008.160454.541437.1U
/userDirectory/deadLetterDirectory/TESTMAIL.SYS00006.Sep302008.160454.541999.1U
```

Figure 57. Code sample

  - If the configured dead letter directory already exists, then CSSMTP uses the existing directory.
  - If the directories cannot be created during parsing of the configuration file, CSSMTP generates a configuration error.

USEREXIT statement

Use the USEREXIT statement to specify whether or not CSSMTP calls the CSSMTP exit program to interrogate data that is sent to CSSMTP from the JES spool data set.

Requirement: You must install a CSSMTP exit program for this function to work correctly. For more information about the CSSMTP exit program, see "CSSMTP user exit version 3" on page 1394.

The user exit value on MODIFY REFRESH or MODIFY USEREXIT command is not changed until the next JES spool file is opened.

Syntax

```
   ┌─USEREXIT──NONE─────────┐
►►─┤                        ├──────────────────────────────────────────────►◄
   └─USEREXIT──┬─NONE─────┬─┘
               ├─VERSION2─┤
               └─VERSION3─┘
```

Parameters

NONE
: The CSSMTP user exit is not called. You can use the commands that are documented in RFC 821 and RFC 2821. This is the default value.

VERSION2
: The CSSMTP user exit that uses the exit facility token name EZBTCPIPSMTPEXIT is called. Using VERSION2 allows only RFC 821 syntax for the mail read from the JES spool data set. The RFC 2821 commands are not accepted. The exit routine for the previous batch SMTP version continues to function.

VERSION3
: The CSSMTP user exit that uses the exit facility token name EZATCPIPCSSMTPV3 is called. Use VERSION3 to read and to process both RFC 821 and RFC 2821 commands from the JES spool data set.

CSSMTP environment variables

The CSSMTP_CODEPAGE_CONFIG environment variable is for Communication Server SMTP (CSSMTP). The variable is used to set the supported EBCDIC single-byte code page used by the configuration file to convert to EBCDIC IBM-1047. The following code pages are supported:

Table 104. Code pages known to work with CSSMTP

Code page	Description
IBM-037	USA, CANADA, BRAZIL, AND COMMON EUROPE
IBM-273	AUSTRIA AND GERMANY
IBM-277	DENMARK NORWAY
IBM-278	FINLAND SWEDEN
IBM-280	ITALIAN
IBM-281	JAPAN
IBM-282	PORTUGAL
IBM-284	SPANISH
IBM-285	UNITED KINGDOM
IBM-297	FRENCH
IBM-500	INTERNATIONAL
IBM-871	ICELAND
IBM-1047	LATIN 1/ OPEN SYSTEM
IBM-1140	COMMON EUROPE ECECP
IBM-1141	AUSTRIA AND GERMANY ECECP

Table 104. Code pages known to work with CSSMTP (continued)

Code page	Description
IBM-1142	DENMARK NORWAY ECECP
IBM-1143	FINLAND SWEDEN ECECP
IBM-1144	ITALIAN ECECP
IBM-1145	SPANISH ECECP
IBM-1146	UNITED KINGDOM ECECP
IBM-1147	FRENCH ECECP
IBM-1148	INTERNATIONAL ECECP
IBM-1149	ICELAND ECECP
4133	USA
4369	AUSTRIA AND GERMANY
4370	BELGIUM
4371	BRAZIL
4373	DENMARK NORWAY
4374	FINLAND SWEDEN
4376	ITALY
4378	PORTUGAL
4380	LATIN
4381	UNITED KINGDOM
4393	FRANCE
4596	LATIN AMERICA
4967	ICELAND
5143	LATIN OPEN SYS
8229	INTERNATIONAL
8692	AUSTRIA AND GERMANY
12788	ITALY
16421	CANADA
16884	FINLAND SWEDEN
20517	PORTUGAL
20980	DENMARK NORWAY
24613	INTERNATIONAL
25076	DENMARK NORWAY
29172	BRAZIL
32805	JAPAN LATIN
33268	UNITED KINGDOM / PORTUGAL
41460	SWISS
45556	SWISS
49652	BELGIUM
53748	INTERNATIONAL
61696	GLOBAL USE
61711	GLOBAL USE

Table 104. Code pages known to work with CSSMTP (continued)

Code page	Description
61712	GLOBAL USE

See the code set converters supplied in *z/OS Support for Unicode: Using Unicode Services*.

Result: This code page is not used for JES spool files. See "TRANSLATE statement" on page 1388 for JES code page information. The default is EBCDIC IBM-1047.

CSSMTP user exit version 3

Use the Communication Server SMTP (CSSMTP) exit to check, and subsequently accept or reject, outbound mail from the JES spool data set. For example, you can code an exit to check the MAIL FROM: string on outbound mail.

CSSMTP uses the Dynamic Exit Facility (CSVDYNEX macro) provided by MVS. See z/OS MVS Programming: Assembler Services Guide for more information.

The USEREXIT statement in the CSSMTP configuration, along with the MODIFY CSSMTP,USEREXIT command, defines which user exit is called by CSSMTP. For compatibility with the SMTP server, VERSION2 is provided. VERSION3 is provided to take advantage of the additional features provided by CSSMTP.

A sample VERSION3 user exit is included in member CSSMTPV3 in SEZAINST. The name of the macro is EZAYSMTP and the macro is located in SEZANMAC.

Table 105 provides a definition for the VERSION2 and VERSION3 user exits.

Table 105. USEREXIT comparisons

Feature	SMTP server exit	VERSION2	VERSION3
Dynamic Exit ExitName	EZBTCPIPSMTPEXIT	EZBTCPIPSMTPEXIT	EZATCPIPCSSMTPV3
Macro	EZBZSMTP.MACRO	EZBZSMTP.MACRO	EZAYSMTP.MACRO
RFC commands supported	RFC 821	RFC 821	RFC 2821
Sample program	SMTPEXIT.SAMPLE	SMTPEXIT	CSSMTPV3
Parameter list variables			
Feature	**SMTP server exit**	**VERSION2**	**VERSION3**
EZBPACTN	1 - 18	1 -18	1 - 20
EABPBUFF	Pointer to data buffer or 0	Pointer to data buffer or 0	Pointer to data buffer or 0
EZBPCNID	1 - 256, or 257	257	257
EZBPDLEN	Length of data buffer or 0	Length of data buffer or 0	Length of data buffer or 0
EZBPIPV4	IP address or 0	Zero	Zero
EZBPTOKN	Zero or pointer to SAF Token	Pointer to SAF token	Pointer to SAF token
EZBPUSER	User token from EZBAINIT call	User token from EZBAINIT call	User token from EZBAINIT call
EZBPVERS	2	2	3
Return codes			

Table 105. USEREXIT comparisons (continued)

Feature	SMTP server exit	VERSION2	VERSION3
Feature	**SMTP server exit**	**VERSION2**	**VERSION3**
EZBRAGN (call exit again)	Return code 0	Return code 0	Return code 0
EZBRACC (do not call exit again)	Return code 4	Return code 4	Return code 4
EZBRREJ (Reject the JES spool file)	Return code 8	Return code 8	Return code 8
EZBRMAIL(Reject a mail from spool file)	NA	NA	Return code 12
Action code (value) passed to exit routines in EZBPACTN			
Feature	**SMTP server exit**	**VERSION2**	**VERSION3**
EZBAINIT (1)	Yes	Yes	Yes
EZBATERM (2)	Yes	Yes	Yes
EZBADATA (3)	Yes	Yes	Yes
EZBAEXPN (4)	Yes	No (command is not implemented)	No (command is not implemented)
EZBAHELO (5)	Yes	Yes	Yes
EZBAHELP (6)	Yes	No (command is not implemented)	No (command is not implemented)
EZBAMAIL (7)	Yes	Yes	Yes
EZBANOOP (8)	Yes	No (command is not implemented)	No (command is not implemented)
EZBAQUEU (9)	Yes	No (command is not implemented)	No (command is not implemented)
EZBAQUIT (10)	Yes	Yes	Yes
EZBARCPT (11)	Yes	Yes	Yes
EZBARSET (12)	Yes	Yes	Yes
EZBATICK (13)	Yes	No (command is not implemented)	No (command is not implemented)
EZBAVERB (14)	Yes	No (command is not implemented)	No (command is not implemented)
EZBAVRFY (15)	Yes	No (command is not implemented)	No (command is not implemented)
EZBADBUF (16)	Yes	Yes	Yes
EZBAEODB (17)	Yes	Yes	Yes
EZBACONN (18)	Yes	Yes	Yes
EZBAEHLO (19)	NA	No	Yes
EZBASTAR (20)	NA	No	Yes

If the USEREXIT statement specifies VERSION2, then only RFC 821 syntax for the mail read from JES spool files is allowed. If the USEREXIT statement specifies VERSION3 or NONE, then RFC 821 and RFC 2821 syntax for the mail read from JES spool file is allowed. See "USEREXIT statement" on page 1391 for more information.

Information about programming the SMTPEXIT using VERSION2 can be found in "Steps for using the SMTP server exits" on page 1319.

CSSMTPV3 is provided as a programming guide to aid in the implementation of the local policies. It can be found in SEZAINST. If using the CSSMTP exit, a name token (EZATCPIPCSSMTPV3) must be established in SYS1.PARMLIB(PROGxx). See z/OS MVS Initialization and Tuning Guide for more information.

You can use the SETPROG EXIT command to activate and deactivate EZATCPIPCSSMTPV3 exit routines. Seez/OS MVS System Commands for more information.

When you design the CSSMTP exit, consider some of the following design points. Code the exit as efficiently as possible; take all efforts to avoid excessive processing or waiting (for example. I/O operations and DNS resolver calls, while within the exit). Efforts to reject mail might be more efficient if extensive scanning of the data portion of the mail message can be avoided. The exit can allow processing to continue or reject the entire mail message and does not have the ability to reject individual segments of a mail message. The mail message contents cannot be changed in any way by the exit. The exit can accept a mail message at any point and disable further exit calls for that mail message. Only commands that are currently implemented by the CSSMTP program and that are syntactically correct are passed to the exit program.

Read and understand RFCs 2505 and RFC 2635 before undertaking this coding effort. More information about CSSMTP commands and standards are documented in RFCs 2821 and 2822.

The CSSMTP dynamically determines if a CSSMTP exit program exists. This determination is based upon the CSSMTP exit program association with the name token EZATCPIPCSSMTPV3 using the MVS SETPROG command. If you determine that the exit program needs to be called to interrogate data coming from the JES spool data set, follow these steps:

This topic describes how to call the exit program to interrogate data coming from the JES spool data set.

Perform the following steps to call the exit program to interrogate data coming from the JES spool data set:

1. Add code to the user exit program so that the program is compatible with the JES connection.

   Rules:

   - The connection ID is always the value 257.
   - The EZBPIPV4 field that represents the remote IP address is always 0 for the JES connection.

   For SAF token information, the EZBPTOKP field contains the address of the token. The SAF token length is 80 bytes and the SAF token version is 1. The SAF token provides information about the submitting user ID and the submitter node of the JES data. This data can be compared to the sender information about the MAIL FROM: string. For more information about what is provided in the SAF token, see the ICHRUTKN information in z/OS Security Server RACF Data Areas.

2. Compile the user exit with the version 3 copy of the EZAYSMTP DSECT. This action recognizes the changes in the parameter list.

   ____

3. Write the exit in Assembler language. You must use standard z/OS Assembler entry and exit linkage. See z/OS MVS Programming: Assembler Services Guide for the linkage conventions.

   ____

A return code of 8 or 12 results in a reply message that is listed in the log and in the error report. The JES spool file is then subject to the action of the BadSpoolDisp statement. See “BADSPOOLFILEID statement” on page 1327 for details. See “Registers at exit” on page 1400 for details about exit return codes.

The user exit is passed by the generated undeliverable mail notification and the error report.

Restriction: This exit must be reentrant and amode 31 in an authorized library.

The exit is invoked with the settings shown in Table 106.

Table 106. CSSMTP user exit settings

Exit	Description
Authorization	Problem state
Dispatchable unit mode	Task
Cross memory mode	PASN=HASN
Amode	31-bit
ASC mode	Primary address space control (ASC) mode
Interrupt status	Enabled for interrupts
Locks	Unlocked
Control parameters	In the caller's primary address space

Registers at entry

On entry to the exit, the register contents are:

Register 0
Used as a work register by the system.

Register 1
Address of the exit's input parameter list. See Table 107 on page 1398.

Registers 2-12
Unassigned.

Register 13
Address of a 36-word save area.

Register 14
Return address.

Register 15
Entry point address of the exit routine.

The exit input parameter list contains the information shown in Table 107 on page 1398. An assembler macro is available that contains a DSECT describing this area.

The name of the macro is EZAYSMTP and the macro is located in SEZANMAC. The macro enables an optional label but has no operands. The macro provides symbolic names for the four return codes and the twenty action codes. The labels are as shown in Table 107.

Table 107. CSSMTP exit input parameter list

Label name	**Width or value**	**Description**	**Notes**
Parameter list variables			
EZAYSMTP		DSECT Name	
EZBPVERS	One fullword	Version number	A word containing the version number three.
EZBPACTN	One fullword	Action code	An action code describing the buffer contents (if any).
EZBPUSER	One fullword	Returned Reg15 of initialization call	Contains the user supplied token from the EZBAINIT call.
EZBPCNID	One fullword	Connection ID	JES spool data always set to 257.
EZBPTOKP	One fullword	Address of SAF (security) token	SAF information is always returned. CSSMTP sets this field to a 31-bit address that points to the SAF (security) token information. See ICHRUTKN
	Two fullwords	Reserved	Reserved
EZBPIPV4	One fullword	Reserved	Always 0.
EZBPDLEN	One fullword	Length of data in buffer	A word containing the actual length of the data in the buffer. If the buffer length is meaningless for the action code, the length is set to 0.
EZBPBUFF	One fullword	Buffer address	A word containing a 31-bit address that points to the actual buffer. If the buffer length is zero, do not use this parameter.
Return codes			
Label name	**Width or value**	**Description**	**Notes**
EZBRAGN	0	Return code to continue	The exit routine is called again.
EZBRACC	4	Return code to accept mail	The exit routine is not called again until the start of the next mail event.

Table 107. CSSMTP exit input parameter list (continued)

Label name	Width or value	Description	Notes
EZBRREJ	8	Return code to reject entire JES spool file	Mail already accepted are processed and sent.
EZBRMAIL	12	Return code to reject the current mail	If a mail is in progress, it is not sent.
Action codes			
Label name	**Width or value**	**Description**	**Notes**
EZBAINIT	1	Initialization call	Buffer is empty, a return token is expected in R15 and saved in the EZBPUSER field.
EZBATERM	2	Termination call	The application shutting is down and exit return code is ignored. This call (and all others) might not occur during abnormal termination.
EZBADATA	3	RFC 2821 DATA command	
EZBAEXPN	4	RFC 2821 EXPN (expand) command	This action code is never passed to the CSSMTP exit
EZBAHELO	5	RFC 2821 HELO (hello) command	
EZBAHELP	6	RFC 2821 HELP command	This action code is never passed to the CSSMTP exit.
EZBAMAIL	7	RFC 2821 MAIL command	
EZBANOOP	8	RFC 2821 NOOP command	This action code is never passed to the CSSMTP exit.
EZBAQUEU	9	RFC 2821 QUEU (queue) command	This action code is never passed to the CSSMTP exit.
EZBAQUIT	10	RFC 2821 QUIT command	
EZBARCPT	11	RFC 2821 RCPT (recipient) command	
EZBARSET	12	RFC 2821 RSET (Reset) command	
EZBATICK	13	IBM SMTP TICK command	This action code is never passed to the CSSMTP exit.

Table 107. CSSMTP exit input parameter list (continued)

Label name	Width or value	Description	Notes
EZBAVERB	14	IBM SMTP VERB command	This action code is never passed to the CSSMTP exit.
EZBAVRFY	15	RFC 821 VRFY command	This action code is never passed to the CSSMTP exit.
EZBADBUF	16	Data buffer	Data buffer of approximately 1024 bytes of data or less. The data buffers are untranslated.
EZBAEODB	17	End of data buffers	End of data marker This is the last chance to reject this mail message.
EZBACONN	18	End of connection	End of file for JES spool data.
EZBAEHLO	19	RFC 2821 EHLO command	
EZBASTAR	20	RFC 2821 STARTTLS command	

There are two control invocations of the CSSMTP user exit. One for initialization and the other for termination. On return from the initialization call, the content of register 15 is treated as a 4-byte user token that is returned on all other exit invocations. See Table 107 on page 1398 for more information. The user token is not used by CSSMTP but is only passed on subsequent calls to allow a re-entrant exit to have static data (using getmain or some other method). Certain data sets might be read during the initialization call and tables of known spamming Internet addresses might be constructed at this time for later use. The termination call allows report generation or any other clean-up activity that the exit might do prior to the stopping of CSSMTP.

Registers at exit

Upon return from Communication Server SMTP (CSSMTP) exit processing, the register contents must be one of the types listed in Table 108.

Table 108. Register contents upon return from CSSMTP exit processing

Register	Contents
0 - 14	Not applicable

Table 108. Register contents upon return from CSSMTP exit processing (continued)

Register	Contents
15	One of the following return codes: **0** EZBRAGN - Continue to call the exit; **4** EZBRACC - Accept the current command or data, but do not call the exit again until the start of the next mail message.; **8** EZBRREJ - Reject the current JES spool file. Any mail already accepted is processed.; **12** EZBRMAIL - Reject the current mail message in progress.

Return codes that are not valid are converted to the value 0, and processing continues as if the return code was EZBRAGN.

The buffer contents for action codes 3 through 15 or 19 through 20 contain the CSSMTP command. See RFC 2821 for exact syntax and format.

Unknown commands are rejected by CSSMTP and the exit is not called.

Guideline: The CSSMTP command can be uppercase, lowercase, or mixed case.

Interaction between CSSMTP and user exit program

While processing a JES spool file, the HELO, EHLO, MAIL, RSET and QUIT commands reset the last return code to EZBRAGN, which allows the DATA, RCPT, data buffer lines, and the end of mail message line to be processed. After a EZBRACC or EZBRMAIL code is returned from the user exit, the exit is not called again until the next command that resets the last return code is processed. If the EZBRREJ code is returned from the user exit, the spool file is not read again.

Table 109 shows the action code and return code results.

Table 109. Action code and return code results

Action (value)	Last return code	RC= EZBRAGN	RC= EZBRACC	RC= EZBRREJ	RC= EZBRMAIL
EZBAINIT (1)	Ignored	Ignored	Ignored	Ignored	Ignored
EZBATERM (2)	Ignored	Ignored	Ignored	Ignored	Ignored
EZBADATA (3)	Call if EZBRAGN	Continued	Continued	End spool file	Reject mail
EZBAEXPN (4)	This command is not implemented				
EZBAHELO (5)	Reset to EZBRAGN	Continued	Continued	End spool file	Ignored
EZBAHELP (6)	This command is not implemented				

Table 109. Action code and return code results (continued)

Action (value)	Last return code	RC= EZBRAGN	RC= EZBRACC	RC= EZBRREJ	RC= EZBRMAIL
EZBAMAIL (7)	Reset to EZBRAGN	Continued	Continued	End spool file	Reject mail
EZBANOOP (8)	This command is not implemented				
EZBAQUEU (9)	This command is not implemented				
EZBAQUIT (10)	Reset to EZBRAGN	Continued	Continued	End spool file	Ignored
EZBARCPT (11)	Call if EZBRAGN	Continue	Continue	End spool file	Reject mail
EZBARSET (12)	Call if EZBRAGN	Continue	Continue	End spool file	Ignored
EZBATICK (13)	This command is not implemented				
EZBAVERB (14)	This command is not implemented				
EZBAVRFY (15)	This command is not implemented				
EZBADBUF (16)	Call if EZBRAGN	Continue	Continue	End spool file	Reject mail
EZBAEODB (17)	Call if EZBRAGN	Continue	Continue	End spool file	Reject mail
EZBACONN (18)	Ignored	Ignored	Ignored	Ignored	Ignored
EZBAEHLO (19)	Reset to EZBRAGN	Continued	Continue	End spool file	Ignored
EZBASTAR (20)	The STARTTLS command is processed.				

Chapter 31. TIMED daemon

The TIMED daemon is used to provide the time in response to UDP requests. TIMED gives the time in seconds since midnight January 1, 1900. You can start TIMED from the z/OS shell or as a started procedure.

This topic contains the following information:
- "Starting TIMED from z/OS"
- "Starting TIMED as a procedure"

Requirement: TCP/IP must be started prior to starting TIMED.

Guideline: TIMED is different from the TIME daemon available as an internal daemon of INETD. INETD cannot be used to start and perform as a listener for TIMED.

Starting TIMED from z/OS

TIMED is installed in the /usr/lpp/tcpip/sbin/ directory.

To start the TIMED server from the command line, type the timed command as follows:

```
timed [-l] [-p port]
```

The following parameters used for the timed command:

-l
Logs all the incoming requests and responses to the system log. Logged information includes the IP address of the requester.

-p port
Uses the specified port. You can specify the port in which requests are to be received.

Starting TIMED as a procedure

The following sample shows how to start TIMED as a procedure:

```
//TIMED    PROC
//*
//*   5694-A01 (C) Copyright IBM Corp. 1997, 2002
//*   Licensed Materials - Property of IBM
//*   This product contains "Restricted Materials of IBM"
//*   All rights reserved.
//*   US Government Users Restricted Rights -
//*   Use, duplication or disclosure restricted by
//*   GSA ADP Schedule Contract with IBM Corp.
//*   See IBM Copyright Instructions.
//*
//* Function: Time server start procedure
//* SMP/E distribution name: EZATTMDP
//*
//TIMED    EXEC PGM=TIMED,REGION=0K,TIME=NOLIMIT,
//         PARM='POSIX(ON),ALL31(ON),TRAP(OFF)/'
//*STEPLIB DD DISP=SHR,DSN=TCP.SEZALOAD,
//*           VOL=SER=,UNIT=
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//         PEND
```

Figure 58. Starting TIMED as a procedure

Chapter 32. SNTP daemon

You can start the SNTP daemon (SNTPD) from the z/OS shell or as a started procedure.

This topic contains the following information:
- "Starting SNTPD from z/OS"
- "Starting SNTPD as a procedure" on page 1406

Rule: TCP/IP must be started prior to starting SNTPD.

SNTPD uses 123 as the default UDP port. When the low number UDP ports are restricted (RESTRICTLOWPORTS parameter was specified on the UDPCONFIG profile statement for the TCP/IP stack), reserve the port used by SNTPD by specifying a PORT profile statement with the port number and the SNTPD started procedure name. To further restrict access to the SNTPD port, specify the SAF parameter on the PORT profile statement. See "PORT statement" on page 257 for more information.

Restriction: SNTPD cannot be started from INETD.

Starting SNTPD from z/OS

SNTPD is installed in the /usr/sbin/sntpd directory.

To start the SNTP server from the z/OS shell command line, type `sntpd &` on the command line. This starts sntpd and sends it to the background.

The following optional parameters are used for the sntpd command:

-d Enables debugging. Debug messages go to syslogd daemon.

-df *pathname*
Enables debugging. Debug messages go to specified file location.

-pf *pathname*
Specifies z/OS UNIX path for process ID file.

-b *nnnnn*
Acts in broadcast mode. This parameter sends local broadcasts on all interfaces every *nnnnn* seconds. The valid range for **-b** is 1 - 16284.

-m *nnnnn*
Acts in multicast mode. Sends multicast updates (TTL=1) on all interfaces every *nnnnn* seconds. The valid range for -m is 1 - 16284.

-s *n* Use *n* as the stratum level in all replies sent by the server. The valid range for *n* is 1 - 15. If **-s** is not specified or an invalid value is specified, the default stratum level of 1 is used. The stratum level indicates the relative accuracy of the local clock compared to the clocks of other SNTP servers in the network. The value 1 is the most accurate and 15 is the least accurate.

Guideline: The SNTP server always responds to client requests (unicast mode), regardless if **-b** or **-m** start options are specified.

Starting SNTPD as a procedure

The following sample [shipped as SEZAINST(SNTPD)] shows how to start SNTPD as a procedure:

```
//SNTPD    PROC
//*
//* Sample procedure for the Simple Network Time Protocol (SNTP)
//*
//* z/OS Communications Server Version 1 Release 13
//* SMP/E Distribution Name: SEZAINST(EZASNPRO)
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               5694-A01
//*               Copyright IBM Corp. 2002, 2011
//*
//* Status:       CSV1R13
//*
//SNTPD    EXEC PGM=SNTPD,REGION=4096K,TIME=NOLIMIT,
//         PARM='-d'
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//CEEDUMP  DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//*
```

Figure 59. Starting SNTPD as a procedure

Chapter 33. Remote execution server

This topic discusses the TSO remote execution server and contains the following information:

- "Remote execution server cataloged procedure (RXPROC)"
- "RXUEXIT user exit sample" on page 1411
- "z/OS remote execution server" on page 1413

The TSO remote execution server enables TSO commands to be submitted from a remote host and executed on z/OS.

Remote execution server cataloged procedure (RXPROC)

The following shows the Remote execution cataloged procedure (RXPROC):

```
//RXSERVE PROC MODULE='RSHD',
//          EXIT=,
//          TSOPROC=IKJACCNT,
//          MSGCLASS=H,
//          TSCLASS=A,
//          MAXCONN=512,
//          PREFIX=,
//          PURGE=Y,
//          IPV6=Y,
//          SECLABEL=Y,
//          TRACE=
//*
//* z/OS Communications Server
//* SMP/E Distribution Name: EZAEB02V
//*
//* Copyright:    Licensed Materials - Property of IBM
//*               "Restricted Materials of IBM"
//*               5650-ZOS
//*               Copyright IBM Corp. 1991, 2013
//*               US Government Users Restricted Rights -
//*               Use, duplication or disclosure restricted by
//*               GSA ADP Schedule Contract with IBM Corp.
//*  Status = CSV2R1
//*
//* Change Activity =
//* CFD List:
//* $01=PN64129 TCPV3R2 941207  JRC: Change defaults and descriptions
//*                                  for MSGCLASS and TSCLASS.
//* $02=PN73459 TCPV3R2 950831  JB:  Correct descriptions and defaults
//*                                  for MSGCLASS and TSCLASS.
//*
//* Flag Reason   Release  Date   Origin    Description
//* ---- -------- -------- ------ --------  -----------------------
//* $J1= D310.37  CSV2R10  990818 IT97HEIN: REXEC enhancemants:
//*                                         added the PREFIX and
//*                                         PURGE option
//* $J2= MV20462  CSV2R10  991209 DINAKAR : Allow JOB parm
//*                                         abbreviations
//* $N1= MO00080  CSV1R4   020211 CHAMBERS: Put JOB parm abbreviations
//*                                         in EXEC statement
//* $Q1= MV27375  D316     021231 CHAMBERS: added the IPV6 and
//*                                         SECLABEL options
//* $V1= D154901  REBASE   111207 VALLER:   Correct PREFIX= comment
//* End CFD List:
//*
//* Supported PARMS (separated by commas - ',') are:
```

```
//*       EXIT=exitmod  - Name of an exit routine to alter JOB and
//*       EX=exitmod      EXEC parameters for submission of TSO batch
//*                       jobs submitted for remote commands.
//*                       EXIT=NOEXIT can be specified when no exit
//*                       is required.
//*       TSOPROC=proc  - The name of the TSO batch procedure. The
//*       TSO=proc        default is IKJACCNT,  and it can be modified
//*                       in the exit routine specified with the EXIT
//*                       parameter.
//*       MSGCLASS=c    - The MSGCLASS parameter for TSO batch jobs
//*       MSG=c           submitted to execute remote commands.
//*                       Specify a HELD class for this parameter.@02A
//*                       The default is H.  The parameter is not @02C
//*                       to be altered by the exit routine.      @01A
//*       TSCLASS=c     - The SYSOUT class for the SYSTSPRT DD for
//*       TSC=c           submitted jobs.  Specify a different    @02C
//*                       class than the MSGCLASS parameter.  The
//*                       default is A.                           @01A
//*       PURGE=c       - The values for purge are Y or N. Y      @J1A
//*       PUR=c           indicates the job output from the jobs  @J1A
//*                       submitted by the server should be       @J1A
//*                       purged immediately after the job        @J1A
//*                       execution and N indicates that the job  @J1A
//*                       outout will be held in the output queue.
//*       IPV6=c        - The values for ipv6  are Y or N. Y      @Q1A
//*                       indicates the server should attempt     @Q1A
//*                       communication over an IPv6 network.     @Q1A
//*                       Specifying N prevents IPv6-only clients @Q1A
//*                       from communicating with this server.    @Q1A
//*                       This is useful for installations that   @Q1A
//*                       have not migrated user exits to         @Q1A
//*                       accomodate IPv6 addresses.              @Q1A
//*       SECLABEL=c    - The values for seclabel are Y or N. Y   @Q1A
//*       SL=c            indicates the server should attempt     @Q1A
//*                       to add a security label (if one exists) @Q1A
//*                       to the job card following the message   @Q1A
//*                       class parameter. Specifying N prevents  @Q1A
//*                       the server from adding a security label @Q1A
//*                       to the job card.                        @Q1A
//*       TRACE=options - The following options are supported:
//*       TR=options      LOG | NOLOG: controls writing trace records
//*                       on SYSPRINT. NOLOG may be abbreviated as
//*                       NOL.
//*                       SEND | NOSEND: controls sending trace
//*                       records to the client. SEND may be
//*                       abbreviated as SEN and NOSEND as NOS.
//*                       CLIENT=client | ALLCLIENTS: selects a client
//*                       host for which trace records are produced,
//*                       or ALLCLIENTS to trace all clients. CLIENT
//*                       may be abbreviated as CLI and ALLCLIENTS as
//*                       ALLC.
//*                       RESET: sets the options to NOLOG,NOSEND,
//*                       ALLCLIENTS. RESET may be abbreviated as RE.
//*                       If more than one option is specified,
//*                       enclose the options in parentheses.
//* These parameters can also be changed with a MODIFY command.
//*
//* The following parameters may not be changed after START.      @J1A
//*       MAXCONN=n     - The maximum number of open sockets at   @J1A
//*       MAX=n           any one time.  Usually each client      @J1M
//*                       requires 2 sockets while                @J1M
//*                       the command is being processed          @J1M
//*                       and the output is being returned. The   @J1M
//*                       default and minimum value is 512. If a  @J1M
//*                       value of less than 512 is specified,    @J1M
//*                       the default will be used.               @J1M
//*       PREFIX=xxxx   - A four-character value to be used as    @J1A
//*       PRE=xxxx       the first four characters in the       @J1A
//*                      jobname of the jobs that are submitted  @J1A
//*                      by the server.                         @J1A
//*                      The remaining characters of the jobname @J1A
//*                      will be a sequential number between 1   @J1A
//*                      and 9999.                              @V1C
//*
//RXSERVE  EXEC PGM=&MODULE,PARM=('EX=&EXIT,TSO=&TSOPROC',        @N1C
//             'MSG=&MSGCLASS,TSC=&TSCLASS',                     @N1C
//             'MAX=&MAXCONN,PRE=&PREFIX,TR=&TRACE',          @J1C@N1C
//             'PUR=&PURGE,IPV6=&IPV6,SL=&SECLABEL'),      @J1A@N1C@Q1C
//             REGION=7500K,TIME=1440
//*
//*       The C runtime libraries should be in the system's link
//*       list or add them to the STEPLIB definition here.  If you
//*       add them to STEPLIB, they must be APF authorized.
//*
//STEPLIB  DD DSN=TCPIP.SEZATCP,DISP=SHR
//*
//*       SYSPRINT contains runtime diagnostics from RSHD. It can be
//*       a data set or SYSOUT.
//*
//SYSPRINT DD SYSOUT=*
//*
//*       The SYSMDUMP DD statement will cause MVS to provide
//*       an IPCS readable dump for ABENDs.
//*
//*SYSMDUMP DD DISP=SHR,DSN=your.dump.data.set
//*
//*       SYSTCPD explicitly identifies which data set is to be
//*       used to obtain the parameters defined by TCPIP.DATA
//*       when no GLOBALTCPIPDATA statement is configured.
//*       See the IP Configuration Guide for information on
//*       the TCPIP.DATA search order.
//*       The data set can be any sequential data set or a member of
//*       a partitioned data set (PDS).
//*
//SYSTCPD  DD DSN=TCPIP.SEZAINST(TCPDATA),DISP=SHR
```

Figure 60. Remote execution cataloged procedure (RXPROC)

Remote execution server parameters

The system parameters required by the Remote Execution server are passed by the PARM operand of the EXEC statement in the Remote Execution cataloged procedure. Update the following parameters as required by your installation:

EX= or EXIT=
Name of a user exit routine to inspect and alter JOB and EXEC parameters prior to submission of TSO batch jobs initiated by remote commands.

IPV6=
Y or N, indicating whether the server should attempt communication over an IPv6 network. If this option is not specified, the server attempts IPv6 communication. Specifying N for this option prevents IPv6-only clients from communicating with this server.

Tip: This option is useful for installations that have not migrated user exits to accommodate IPv6 addresses.

PRE= or PREFIX=
A four-character value used as the first four characters in the job name of jobs that are submitted. The remaining characters of the job name is a sequential number in the range of 1 - 9999.

PUR= or PURGE=
Y or N, indicating whether a job submitted by the server should be purged immediately after execution or held in the output queue.

TSO= or TSOPROC=
The name of the TSO batch procedure. The default is IKJACCNT. The name IKJACCNT can be modified in the exit routine specified with the EXIT parameter.

MSG= or MSGCLASS=
The MSGCLASS parameter for TSO batch jobs submitted to execute remote commands. The default is H, which points to a HELD output class.

Restrictions:

- This parameter must not be altered by the exit routine.
- For JES3 users, the HELD output class needs to be defined as a HELD output class for external writer.

TSC= or TSCLASS=
The SYSOUT class for the SYSTSPRT DD statement for submitted jobs. The default is A.

Restriction: For JES3 users, the HELD output class needs to be defined as a HELD output class for external writer.

MAX= or MAXCONN=
The maximum number of open sockets at any one time. Usually, each client requires 2 sockets while the command is being processed and the output is being returned. The minimum acceptable value is 512. This is also the default.

TR= or TRACE=
The trace options that are to be in effect for the Remote Execution server.

Rule: If more than one trace parameter is specified, enclose the parameters in parentheses.

LOG
Specifies trace records written to SYSPRINT.

NOL= or NOLOG
Specifies no trace records written to SYSPRINT.

SEN= or SEND
Specifies trace records sent to the client.

NOS= or NOSEND
Specifies no trace records sent to the client.

CLI= or CLIENT=*client*
Specifies a specific client host for which trace records are to be produced.

ALLC= or ALLCLIENTS
Specifies that trace records are to be produced for all clients.

RE= or RESET
Sets the trace options to NOLOG, NOSEND, ALLCLIENTS.

SL= or SECLABEL=
Y or N, indicating whether the server should attempt to add a security label to

the job card. If this option is not specified, the server attempts to add a security label to the job card. If Y is specified for this option, the server adds a security label (if one exists) to the job card following the message class parameter. For more information about the multilevel security environment and configuring z/OS Communications Server in that environment, see the multilevel security information in the z/OS Communications Server: IP Configuration Guide.

Use the MODIFY command to dynamically change all but the following parameters:
- MAXCONN
- PREFIX
- IPv6
- SECLABEL

Tip: All parameters can now be abbreviated. For example, EXIT can be abbreviated to EX.

RXUEXIT user exit sample

The following user exit is shipped as a sample in the RXUEXIT member of the SEZAINST data set:

```
*********************************************************************
*
*            Communications Server IP
*
* Name:      RXUEXIT
*
* SMP/E Distribution Name: EZAEBRXU
*
* Function:   This exit will add a CLASS parameter to the JOB
*             statement submitted by the REXECD server in CS/390.
*
*
* Copyright:    Licensed Materials - Property of IBM
*
*               "Restricted Materials of IBM"
*
*               5694-A01
*
*               Copyright IBM Corp. 1977, 2008
*
*               US Government Users Restricted Rights -
*               Use, duplication or disclosure restricted by
*               GSA ADP Schedule Contract with IBM Corp.
*
* Status:       CSV1R10
*
*
* Interface:  Parameter list (R1 points to it on entry):
*
*        +0:  A pointer to a mixed AF_INET or AF_INET6 address
*             structure
*             +0: (2 bytes) Address Family AF_INET, or AF_INET6.
*             +2: (2 bytes) Server port.
*             +4: (4 or 16 bytes) Client AF_INET or AF_INET6 IP
*                 address.
*
*        +4:  Pointer to JOB card parameters (up to 1024
*             characters, terminated by a X'00').  This may be
*             modified by the exit routine.  It is set to
*             "userid USER=userid,[PASSWORD=passwd,
*             ]MSGCLASS=msgclass" at entry,  where 'msgclass'
*             is as specified in the deamon parameters,  and
*             and 'userid' and 'passwd' are as received from
*             the client.
*
*        +8:  Pointer to EXEC card parameters (up to 256
*             characters, terminated by a X'00').  This may be
*             modified by the exit routine.  It is set to the
*             procedure specified on the PROC parameter, or the
*             default IKJACCNT, at entry.
*
*        +C:  Pointer to JES control buffer (up to 256 characters,
*             terminated by a X'00').  This may be modified by the
*             exit routine.  The contents are inserted into the
*             JCL stream following the JOB card and before the
*             EXEC card.
*
*             The EXEC and JES buffer contents are written
*             directly to the internal reader without being parsed.
*             The buffer contents  must provide line seperation by
*             including a NL or CRLF as required.
*
* Logic:      The typical contents of the JOB statement buffer
*             are; userid,USER=userid,PASSWORD=password,
*             MSGCLASS=H,SECLABEL=seclabel
*
*             The JOB statement buffer is 1024 bytes in length.
*             The contents of the buffer are null terminated.
*             If the buffer contents are altered, the user must
*             ensure they are null terminated (one byte x'00')
*             and that the total length including termination
*             byte does not exceed the buffer length.
*
*             The JES control statement buffer is 256 bytes in
*             length and the contents are null terminated.
*
* Abends:     - none -
*
* Returncode: RC = 0
*
*********************************************************************
PARMS    DSECT
PTRINET  DC    F'0'                *-> AF-INET or AF-INET6 socket
*                                      address
PTRJOBP  DC    F'0'                *-> Job statement parameters
PTREXECP DC    F'0'                *-> EXEC statement parameters
PTRJES   DC    F'0'                *-> JES control buffer
*
BUFSIZE  EQU   1024                *JOB statement buffer size
*
.* RXUEXIT INIT   'REXECD add class parameter to JOB statement'
.* *
RXUEXIT  CSECT                     Establish the RXUEXIT csect
RXUEXIT  AMODE 31
RXUEXIT  RMODE ANY
         USING RXUEXIT,12          Establish code addressability
         STM   14,12,12(13)        Save the caller's registers
         LR    12,15               Setup the local base register
         LR    2,1                 *Parm pointer
         USING PARMS,2             *Parameter addressability
         L     4,PTRJOBP           *-> Job card parameters
         LR    5,4                 *-> Start of buffer
         LA    6,1                 *Scan 1 byte at a time
         LA    7,BUFSIZE(5)        *-> First byte after buffer
         BCTR  7,0                 *-> Last byte to scan
SCANLOOP EQU   *
```

```
         CLI   0(5),0              *Is this string termination ?
         BE    GOTEND              *- Yes
         BXLE  5,6,SCANLOOP        *Continue scan for term
* ----------------------------------------------------------------
* If string is not null terminated, return without altering
* ----------------------------------------------------------------
         B     RETURN              *Should not happen.
GOTEND   EQU   *
         LR    6,5                 *address of null byte
         SR    5,4                 *L'job parameter statements
         LA    5,L'CLASS(5)        *New parameter length
         CH    5,=AL2(BUFSIZE)     *Do we exceed buffer size?
         BNH   LENOK               *- No, there is room enough
* ----------------------------------------------------------------
* String length would exceed buf size so return without altering
* ----------------------------------------------------------------
         B     RETURN              *Return without modification
*
LENOK    EQU   *
         MVC   0(L_CLASS,6),CLASS  *Move class statement to buff
         L     6,PTRJES            *Get address of JES buffer
         MVC   0(L_JES2,6),JES2CNTL *Move JES2 control to buffer
RETURN   EQU   *
         LM    14,12,12(13)        Restore the registers
         LA    15,0(0,0)           Load the return code
         BR    14                  Return
*
         LTORG
*
CLASS    DC    C',CLASS=A'         *Class statement
         DC    X'00'               *null termination byte
L_CLASS  EQU   *-CLASS
*
JES2CNTL DC    C'/*JOBPARM SYSAFF=ANY' *JES2 system affinity
         DC    X'00'                    *null termination byte
L_JES2   EQU   *-JES2CNTL
*
JES3CNTL DC    C'//*MAIN SYSTEM=(MAIN1)' *JES3 main assignment
         DC    X'00'                      *null termination byte
L_JES3   EQU   *-JES3CNTL
*
         END
```

Figure 61. RXUEXIT user exit

z/OS remote execution server

The z/OS UNIX System Services remote execution servers, orexecd and orshd, allow UNIX commands to be submitted from a remote host and executed on z/OS.

z/OS UNIX System Services REXECD command (orexecd)

The following syntax is used in the /etc/inetd.conf file to define the arguments used to invoke the orexecd command:

```
►►──orexecd──┬────┬──┬────┬──┬────┬──┬────┬──┬────┬──────────────────►◄
             └─-d─┘  └─-l─┘  └─-v─┘  └─-c─┘  └─-s─┘
```

The following options are supported:

-d Print debug information to syslogd.

-l Write each successful login to syslogd with the remote user, remote system, local user, and the command executed.

-v Write the title and ptf level to syslogd.

-c Write all messages in uppercase.

-s Invoke the remote shell as a login shell (that is, run /etc/profile and $HOME/.profile).

z/OS UNIX System Services RSHD command (orshd)

The following command is used in the /etc/inetd.conf file to define the arguments used to invoke orshd:

```
►►──orshd──┬────┬──┬────┬──┬────┬──┬────┬──┬────┬──┬────┬──┬────┬────────────────►
           └─-a─┘  └─-d─┘  └─-l─┘  └─-v─┘  └─-c─┘  └─-r─┘  └─-s─┘

►──┬──────────────┬──┬────┬──┬────┬──┬────┬──┬────┬─────────────────────────────►◄
   └─-k mechanism─┘  └─-e─┘  └─-m─┘  └─-i─┘  └─-t─┘
```

The following options are supported:

-a Look up host name and check that the address and host name correspond.

-d Print debug information to syslogd.

-l Write each successful login to syslogd with the remote user, remote system, local user, and the command executed.

-v Write the title and ptf level to syslogd.

-c Write all messages in uppercase.

-r If a client passes a null password, invoke the /usr/sbin/ruserok user exit to authenticate the user ID.

-s Invoke the remote shell as a login shell (that is, run /etc/profile and $HOME/.profile).

-k mechanism
Specifies the authentication mechanism to be used to authenticate the client. Valid values for mechanism are KRB5 and GSSAPI.

-e Requires the client to encrypt the connection.

-m Require Kerberos5 clients to present a cryptographic checksum of initial connection information, such as the name of the user that the client is trying to access in the initial authenticator. This checksum provides additional security by preventing an attacker from changing the initial connection information. If this option is specified, older Kerberos5 clients that do not send a checksum in the authenticator is not able to authenticate to this server. This option is mutually exclusive with the **-i** option and is only valid if **-k KRB5** is specified.

If neither the **-m** or **-i** options are specified, checksums are validated if presented. Because it is difficult to remove a checksum from an authenticator without making the authenticator invalid, this default mode is almost as significant of a security improvement as **-m** if new clients are used. It has the additional advantage of backwards compatibility with some clients. Clients before Kerberos V5, Beta5, generate invalid checksums; if these clients are used, the **-i** option must be used.

-i Ignore authenticator checksums if provided. This option ignores authenticator checksum presented by current Kerberos clients to protect initial connection information; it is the opposite of **-m**. This option is provided because some older clients (particularly clients predating the release of Kerberos V5 Beta5, May 1995) present invalid checksums that prevent Kerberos authentication from succeeding in the default mode. This option is mutually exclusive with the **-m** option and is only valid if **-k KRB5** is specified.

-t Use this option to set the KRB5_SERVER_KEYTAB environment variable. If this environment variable is set, the Security Runtime uses a local instance of the Kerberos security server to decrypt service tickets instead of obtaining the key from a key table.

Requirement: The orshd application must have at least read access to the IRR.RUSERMAP resource in the FACILITY class in order to use this capability. For more information, see z/OS Integrated Security Services Network Authentication Service Administration.

RSHD command (orshd) environment variables

Table 110 provides a list of environment variables used by RSHD command () that can be tailored to a particular installation.

Table 110. RSHD command (orshd) environment variables

Environment variable	Server, Client or Command type application	Description	Specific coding rules (or a link to syntax)
_EUV_SVC_DBG_FILENAME	ORSHD	Specifies the fully qualified name of the file that receives debug messages. Debug messages are written to the file specified by the _EUV_SVC_STDOUT _FILENAME if this environment variable is not defined. If _EUV_SVC_STDOUT _FILENAME is not specified, debug messages are written to stdout.	Specifying debug (-d or -D option) and authentication (-k or -K) sets the variable to the value /etc/skrb/krb.log. If you want to allow the user-specified variable to be used, do not specify debug (-d or -D option).
_EUV_SVC_DBG_MSG_LOGGING	ORSHD	Specifies whether authentication trace records are generated when authentication is requested (-k or -K option). These trace records are stored in an internal wrap table. The following values can be specified: • 0 means do not generate trace records (this is the default) • 1 means generate trace records	Specifying debug (-d or -D option) and authentication (-k or -K) sets the variable value to 1. If you want to allow the user-specified variable to be used, do not specify debug (-d or -D option).

Table 110. RSHD command (orshd) environment variables (continued)

Environment variable	Server, Client or Command type application	Description	Specific coding rules (or a link to syntax)
_EUV_SVC_DBG_TRACE	ORSHD	Specifies whether authentication trace records are generated when authentication is requested (-k or -K option). These trace records are stored in an internal wrap table. The following values can be specified: • 0 means do not generate trace records (this is the default) • 1 means generate trace records	Specifying debug (-d or -D option) and authentication (-k or -K) sets the variable value to 1. If you want to allow the user-specified variable to be used, do not specify debug (-d or -D option).
KRB5_SERVER_KEYTAB	ORSHD	Specifies whether the Security Runtime uses a local instance of the Kerberos security server to decrypt service tickets instead of obtaining the key from a key table. The default is to obtain the key from a key table. • 0 means obtain the key from a key table • 1 means use a local instance .	Specifying the -t or -T option sets the variable to 1.

Appendix A. Translation tables

This topic contains the following information:

TCP/IP Services uses translation tables to convert transmitted data from EBCDIC to ASCII. Because these tables do not always include all the desired characters, TCP/IP allows you to create and customize tables without having to recompile source code. Translation tables are stored in binary form on disk. TCP/IP provides standard tables that are used as the default if you do not customize your own.

TCP/IP Services used the following types of translation tables:

- Single-byte character set (SBCS) translation tables are used for single-byte characters.
- Double-byte character sets (DBCS) translation tables are used for converting double-byte characters. DBCS translation tables are required for character sets such as Japanese Kanji, which contains too many characters to represent using single-byte codes. SBCS translation tables provide mappings for a maximum of 256 characters. DBCS translation tables can provide up to a theoretical maximum of 65 535 character mappings; however, DBCS character sets usually contain less than this number.

The FTP program provides an FTP.DATA statement, DUMP 81, that is available for tracing the translate tables. When set on, 256 bytes of each translate table can be traced as follows:

- When the FTP STAT command is entered, the translate tables being used by the server for the control and data connection is traced. When the FTP LOCSTAT command is entered, the translate tables being used by the client is traced.
- When an FTP command is entered to change the translate table, the new table is traced. If the change is entered as a locsite command then it is traced by the client. If the change is entered as a site command, then it is traced by the server.

  See z/OS Communications Server: IP User's Guide and Commands FTP DUMP command information for instructions on entering the command or adding it to FTP.DATA for the server and client. For all trace entries, if the trace is on the server side, it is written to syslogd. Where the client trace entries are written depends on how the client is run. If it is TSO interactive or run in OEM, the trace appears on the console. If it is a batch job, it is in the job output, and if it is being used by Rexx, it is in the Rexx output.

The FTP program also provides a multi-byte character set (MBCS) to support the Chinese standard GB18030. This support is provided by **iconv** with code page IBM-5488; it does not allow for customized tables. See Chapter 18, "File Transfer Protocol," on page 697 for a description of the ENCODING and MBDATACONN statements that define this support for FTP. Also, see z/OS Communications Server: IP User's Guide and Commands for information about how to use the LOCSITE and SITE subcommands to specify ENCODING and MBDATACONN values.

The following topics describe how to create and customize both SBCS and DBCS translation tables and explain how they are used by the programs in TCP/IP Services.

SBCS translation table hierarchy

Different programs look for special translation tables to use. The program chooses one of the customized tables, as described in Table 111. The program first searches for a customized table that you have built. If the program fails to find one of the customized tables, it uses the default table supplied in *hlq*.STANDARD.TCPXLBIN. Table 111 provides the customized translation tables and default table names for the programs.

Guideline: FTP server and FTP client optionally use **iconv** instead of the external tables for single-byte conversion. The use of **iconv** is specified in FTP.DATA or by SITE/LOCSITE commands.

Table 111. SBCS translation table hierarchy

Program	Customized translation tables	Default translation table
FTP client control connection when no TRANSLATE parameter is specified on the ftp command	1. EXTENSIONS UTF8 2. Data set specified in the CTRLCONN configuration statement in FTP.DATA 3. Data set specified in the CCTRANS configuration statement in FTP.DATA 4. user_id.FTP.TCPXLBIN 5. *hlq*.FTP.TCPXLBIN 6. user_id.STANDARD.TCPXLBIN 7. hlq.STANDARD.TCPXLBIN 8. 7-bit ASCII - (ISO8859-1 for the network code page and IBM-1047 for the file system code page) 9. FTP internal 7-bit tables	The default should be number 8 for most all cases.
FTP client data connection when no TRANSLATE parameter is specified on the ftp command	1. Data set specified in the SBDATACONN configuration statement in FTP.DATA 2. Data set specified in the SBTRANS configuration statement in FTP.DATA 3. user_id.FTP.TCPXLBIN 4. hlq.FTP.TCPXLBIN 5. user_id.STANDARD.TCPXLBIN 6. hlq.STANDARD.TCPXLBIN 7. The same translation tables established for the control connection	The default is number 7.
FTP client when TRANSLATE parameter is specified on the ftp command for both control connections and data connections[1]	1. If the client is started in the z/OS UNIX System Services shell: $HOME/data_set.tcpxlbin 2. user_id.data_set.TCPXLBIN 3. *hlq*.data_set.TCPXLBIN	There is no default.

Table 111. SBCS translation table hierarchy (continued)

Program	Customized translation tables	Default translation table
FTP server control connections	1. EXTENSIONS UTF8 2. Data set specified in the CTRLCONN configuration statement in FTP.DATA 3. Data set specified in the CCXLATE configuration statement in FTP.DATA 4. jobname.SRVRFTP.TCPXLBIN 5. hlq.SRVRFTP.TCPXLBIN 6. jobname.STANDARD.TCPXLBIN 7. *hlq*.STANDARD.TCPXLBIN 8. 7-bit ASCII - (ISO8859-1 for the network code page and IBM-1047 for the file system code page) 9. FTP internal 7-bit tables	The default should be number 8 for most all cases.
FTP server data connections	1. Data set specified with DD: SYSFTSX in the FTP start procedure 2. Data set specified in the SBDATACONN configuration statement in FTP.DATA 3. Data set specified in the XLATE configuration statement in FTP.DATA 4. jobname.SRVRFTP.TCPXLBIN 5. hlq.SRVRFTP.TCPXLBIN 6. jobname.STANDARD.TCPXLBI 7. *hlq*.STANDARD.TCPXLBIN 8. The same translation tables established for the control connection	The default is number 8.
LPR Client	1. *user_id*.LPR.TCPXLBIN 2. *hlq*.LPR.TCPXLBIN 3. *user_id*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
LPR Client (TRANSLATE)	1. *user_id.data_set*.TCPXLBIN 2. *hlq.data_set*.TCPXLBIN 3. *user_id*.LPR.TCPXLBIN 4. *hlq*.LPR.TCPXLBIN 5. *user_id*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
LPD Server	*jobname.data*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
LPD Server (TRANSLATE)	1. *jobname.data_set*.TCPXLBIN 2. *hlq.data_set*.TCPXLBIN	None. Printer services cannot be used.
PORTMAP	1. *user_id*.STANDARD.TCPXLBIN 2. *jobname*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
REXEC	*user_id*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN
SMTP	1. *jobname.SMTP*.TCPXLBIN 2. *hlq.SMTP*.TCPXLBIN 3. *jobname*.STANDARD.TCPXLBIN	*hlq*.STANDARD.TCPXLBIN

Table 111. SBCS translation table hierarchy (continued)

Program	Customized translation tables	Default translation table
Telnet Client	1. *user_id*.TELNET.TCPXLBIN 2. *hlq*.TELNET.TCPXLBIN 3. *user_id*.STANDARD.TCPXLBIN	*hlq*.TELNET.TCPXLBIN
Telnet Client (TRANSLATE)	1. *user_id.data_set*.TCPXLBIN 2. *hlq.data_set*.TCPXLBIN	None. Program Halts.

Notes:

1. *jobname* is the name specified either on the PROC or JOB statement.
2. *user_id* is the ID of the user who issued the command.
3. *data_set* is the name entered on the TRANSLATE parameter for the program. See z/OS Communications Server: IP User's Guide and Commands for information about specifying the TRANSLATE parameter for the required program.
4. High Level Qualifier (*hlq*) specified in the TCPIP.DATA configuration statement, DATASETPREFIX.

The Telnet client requires translation tables that are different from the default table *hlq*.STANDARD.TCPXLBIN. Customized translation tables for Telnet clients are provided in the install libraries as *hlq*.TELNET.TCPXLBIN and *hlq*.TELNETSE.TCPXLBIN. If these data sets are not found, the Telnet client uses the default table.

Telnet (for Linemode) uses **iconv** services with the CODEPAGE statement in the TELNETPARMS block to specify country and region translation tables. TCPXLBIN translation tables are not used. If CODEPAGE is in error or not specified, see "CODEPAGE statement" on page 596for default values used. If custom code pages are required, see the information about globalization in the z/OS XL C/C++ Programming Guide for details about how to create your own conversions.

Customizing SBCS translation tables

All SBCS translation table members contain two tables. The first table is used to translate from ASCII to EBCDIC. The second table is used to translate from EBCDIC to ASCII.

To read the translation tables, find the row for the first hex digit (**1**) and the column for the second hex digit (**2**). The point where the row and column intersect is the translation value.

For example, to find the EBCDIC translation for the ASCII character A7, find row A0 (**3**) and column 07 (**4**) in "ASCII-to-EBCDIC table" on page 1421. The point where row A0 and column 07 intersect shows a value of X'7D', so the ASCII character X'A7' is translated to X'7D' in EBCDIC.

To customize the translation table, alter the translate value where the row and column intersect to the new value.

You can edit and modify translation table members in the SEZATCPX data set.

1. Do not use the TRANSLATE option for the FTP client if the SBCS table you need for data transfer does not support standard encodings for the portable character set. Such a translation table can adversely affect the EBCDIC to ASCII conversion of commands sent over the control connection.

ASCII-to-EBCDIC table

The samples shown in Figure 62 and Figure 63 are shipped as the *hlq*.STANDARD.TCPXLBIN compiled translation table, and unless modified, is used for EBCDIC-to-ASCII or ASCII-to-EBCDIC translation.

```
;
; ASCII-to-EBCDIC table
;                      4                    2
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;                                                      1
  00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F    ; 00 ;
  10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F    ; 10 ;
  40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61    ; 20 ;
  F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F    ; 30 ;
  7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6    ; 40 ;
  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D    ; 50 ;
  79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96    ; 60 ;
  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07    ; 70 ;
  00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F    ; 80 ;
  10 11 12 13 3C 3D 32 26 18 19 3F 27 22 1D 35 1F    ; 90 ;
  40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61    ; A0 ;  ←3
  F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F    ; B0 ;
  7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6    ; C0 ;
  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 AD E0 BD 5F 6D    ; D0 ;
  79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96    ; E0 ;
  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07    ; F0 ;
```

Figure 62. ASCII-to-EBCDIC translation table

EBCDIC-to-ASCII table

```
;
; EBCDIC-to-ASCII table
; 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
;
  00 01 02 03 1A 09 1A 7F 1A 1A 1A 0B 0C 0D 0E 0F    ; 00 ;
  10 11 12 13 1A 0A 08 1A 18 19 1A 1A 1C 1D 1E 1F    ; 10 ;
  1A 1A 1C 1A 1A 0A 17 1B 1A 1A 1A 1A 1A 05 06 07    ; 20 ;
  1A 1A 16 1A 1A 1E 1A 04 1A 1A 1A 1A 14 15 1A 1A    ; 30 ;
  20 A6 E1 80 EB 90 9F E2 AB 8B 9B 2E 3C 28 2B 7C    ; 40 ;
  26 A9 AA 9C DB A5 99 E3 A8 9E 21 24 2A 29 3B 5E    ; 50 ;
  2D 2F DF DC 9A DD DE 98 9D AC BA 2C 25 5F 3E 3F    ; 60 ;
  D7 88 94 B0 B1 B2 FC D6 FB 60 3A 23 40 27 3D 22    ; 70 ;
  F8 61 62 63 64 65 66 67 68 69 96 A4 F3 AF AE C5    ; 80 ;
  8C 6A 6B 6C 6D 6E 6F 70 71 72 97 87 CE 93 F1 FE    ; 90 ;
  C8 7E 73 74 75 76 77 78 79 7A EF C0 DA 5B F2 F9    ; A0 ;
  B5 B6 FD B7 B8 B9 E6 BB BC BD 8D D9 BF 5D D8 C4    ; B0 ;
  7B 41 42 43 44 45 46 47 48 49 CB CA BE E8 EC ED    ; C0 ;
  7D 4A 4B 4C 4D 4E 4F 50 51 52 A1 AD F5 F4 A3 8F    ; D0 ;
  5C E7 53 54 55 56 57 58 59 5A A0 85 8E E9 E4 D1    ; E0 ;
  30 31 32 33 34 35 36 37 38 39 B3 F7 F0 FA A7 FF    ; F0 ;
```

Figure 63. EBCDIC-to-ASCII translation table

Syntax rules for SBCS translation tables

The following syntax rules apply to SBCS translation tables:

- Blanks are used only as delimiters for readability purposes.
- Information to the right of a semicolon (**;**) is a comment.

SBCS country or region translation tables

Rather than customize the table in SEZATCPX(STANDARD), you can use the following translation table members, which are included with the code.

These translation table members are in the same format as that used in SEZATCPX(STANDARD). To use these table members, you must convert them to binary format using CONVXLAT and store the resulting binary tables in an appropriate data set within the SBCS translation table hierarchy. (See "SBCS translation table hierarchy" on page 1418.) For more information about using the TSO CONVXLAT command, see "Using TSO CONVXLAT to convert translation tables to binary" on page 1428.

The editable translation tables used by the Telnet client application are members of the SEZATELX data set and are derived from the identified code pages. The editable tables used by other applications, such as FTP, are members of the SEZATCPX data set.

The following members can be used by both Telnet client and non-Telnet SBCS applications such as FTP, SMTP, and so on. They are found in both SEZATELX and SEZATCPX.

Restriction: Identically named members in the two data sets are **not** the same.

Table 112. Translation table members for Telnet client and non-Telnet SBCS applications

Member name	Description	Code page
AUSGER *	Austrian-German code page	850<->273
BELGIAN *	Belgian code page	850<->500
CANADIAN *	Canadian code page	850<->037
CUSTOM *	Code page	819<->1047
DANNOR *	Danish-Norwegian code page	850<->277
DUTCH *	Dutch code page	850<->037
EAUSGER	Austrian-German with Euro support	858<->1141
EBELGIAN	Belgian with Euro support	858<->1148
ECANADIAN	Canadian with Euro support	858<->1140
EDANNOR	Danish-Norwegian with Euro support	858<->1142
EDUTCH	Dutch with Euro support	858<->1140
EFINSWED	Finnish-Swedish with Euro support	858<->1143
EFRENCH	French with Euro support	858<->1147
EITIALIAN	Italian with Euro support	858<->1144
EPORTUGU	Portuguese with Euro support	858<->1140
ESPANISH	Spanish with Euro support	858<->1145
ESWISFRE	Swiss-French with Euro support	858<->1148
ESWISGER	Swiss-German with Euro support	858<->1148
EUK	United Kingdom with Euro support	858<->1146
EUS	United States with Euro support	858<->1140
FINSWED *	Finnish-Swedish code page	850<->278
FRENCH *	French code page	850<->297

Table 112. Translation table members for Telnet client and non-Telnet SBCS applications (continued)

Member name	Description	Code page
ITALIAN *	Italian code page	850<->280
JAPANESE *	Japanese code page	850<->281
JPNALPHA	Japanese Code code page	1041<->029
JPNKANA	Japanese Code code page	1041<->102
KOR0891	Korean Code code page	0891<->083
KOR1088	Korean Code code page	1088<->083
PORTUGUE *	Portuguese code page	850<->037
PRC1115	People's Republic of China code page	1115<->083
SPANISH *	Spanish code page	850<->284
SWISFREN *	Swiss-French code page	850<->500
SWISGERM *	Swiss-German code page	850<->500
TAI0904	Taiwan code page	0904<->003
TAI1114	Taiwan code page	1114<->003
UK *	United Kingdom code page	850<->285
US *	United States code page	850<->037

Note: See “ISO-8 and IBM PC interpretations for ASCII and EBCDIC code points” on page 1424.

The following SBCS translation table members are only used by Telnet 3270 DBCS Transform support.

Restriction: These are found only in SEZATELX.

Table 113. SBCS translation table members for Telnet 3270 DBCS transform support

Member name	Description	Code page
A8E	Japanese 8-bit English	0819<->1027
A8K	Japanese 8-bit Katakana	0819<->0290
J8E	Japanese JIS 8-bit English	Unassigned
J8K	Japanese JIS 8-bit Katakana	Unassigned
SJDCE	Japanese DEC English	Unassigned
SJDCK	Japanese DEC Katakana	Unassigned
SJECE	Japanese Extended Unix English JIS	X0201<->1027
SJECK	Japanese Extended Unix Katakana JIS	X0201<->0290
KOR0891	Korean code page	0891<->0833
KOR1088	Korean code page	1088<->0833
PRC1115	People's Republic of China code page	1115<->0836
TAI0904	Taiwan code page	0904<->0037
TAI1114	Taiwan code page	1114<->0037

ISO-8 and IBM PC interpretations for ASCII and EBCDIC code points

The tables in the SEZATCPX data set use the ISO-8 interpretations for certain ASCII code points. These code points are mapped to EBCDIC code points, as shown in Table 114.

Table 114. ISO-8 interpretations for certain ASCII and EBCDIC code points

ASCII code point	EBCDIC code point	ISO-8 interpretation
X'1A'	X'3F'	SUB (substitution character)
X'1C'	X'1C'	IFS (interchange file separator)
X'7F'	X'07'	DEL (delete character)

If you want to use IBM PC interpretations for these code points, you can modify your table, as shown in Table 115:

Table 115. IBM PC interpretations for certain ASCII and EBCDIC code points

ASCII code point	EBCDIC code point	IBM PC interpretation
X'1A'	X'1C'	IFS (interchange file separator)
X'1C'	X'07'	DEL (delete character)
X'7F'	X'3F'	SUB (substitution character)

DBCS translation table hierarchy

Table 116 describes the search order used by certain programs when they are configured to load one or more DBCS translation tables.

If the customized DBCS translation tables are not found, then the default table data sets provided with the install libraries are used. If the default tables cannot be read, then error messages are issued, and the required DBCS conversion is unavailable for the program.

Table 116. DBCS translation table hierarchy

Program	Option	Customized translation tables	Default translation table
FTP client	Hangeul	1. *user_id*.FTP.TCPHGBIN 2. *hlq*.FTP.TCPHGBIN 3. *user_id*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
FTP client	Kanji	1. *user_id*.FTP.TCPKJBIN 2. *hlq*.FTP.TCPKJBIN 3. *user_id*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
FTP client	SChinese	1. *user_id*.FTP.TCPSCBIN 2. *hlq*.FTP.TCPSCBIN 3. *user_id*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
FTP client	TChinese	1. *user_id*.FTP.TCPCHBIN 2. *hlq*.FTP.TCPCHBIN 3. *user_id*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN

Table 116. DBCS translation table hierarchy (continued)

Program	Option	Customized translation tables	Default translation table
FTP client	Hangeul and TRANSLATE *	1. *user_id.data_set*.TCPHGBIN 2. *hlq.data_set*.TCPHGBIN 3. *user_id*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
FTP client	Kanji and TRANSLATE *	1. *user_id.data_set*.TCPKJBIN 2. *hlq.data_set*.TCPKJBIN 3. *user_id*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
FTP client	SChinese and TRANSLATE *	1. *user_id.data_set*.TCPSCBIN 2. *hlq.data_set*.TCPSCBIN 3. *user_id*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
FTP client	TChinese and TRANSLATE *	1. *user_id.data_set*.TCPCHBIN 2. *hlq.data_set*.TCPCHBIN 3. *user_id*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN
FTP Server	Hangeul	1. *jobname*.SRVRFTP.TCPHGBIN 2. *hlq*.SRVRFTP.TCPHGBIN 3. *jobname*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
FTP Server	Kanji	1. *jobname*.SRVRFTP.TCPKJBIN 2. *hlq*.SRVRFTP.TCPKJBIN 3. *jobname*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
FTP Server	SChinese	1. *jobname*.SRVRFTP.TCPSCBIN 2. *hlq*.SRVRFTP.TCPSCBIN 3. *jobname*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
FTP Server	TChinese	1. *jobname*.SRVRFTP.TCPCHBIN 2. *hlq*.SRVRFTP.TCPCHBIN 3. *jobname*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN
LPR Client	Hangeul	1. *user_id*.LPR.TCPHGBIN 2. *hlq*.LPR.TCPHGBIN 3. *user_id*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
LPR Client	Kanji	1. *user_id*.LPR.TCPKJBIN 2. *hlq*.LPR.TCPKJBIN 3. *user_id*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
LPR Client	SChinese	1. *user_id*.LPR.TCPSCBIN 2. *hlq*.LPR.TCPSCBIN 3. *user_id*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
LPR Client	TChinese	1. *user_id*.LPR.TCPCHBIN 2. *hlq*.LPR.TCPCHBIN 3. *user_id*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN
LPD Server	Hangeul	1. *jobname*.LPD.TCPHGBIN 2. *hlq*.LPD.TCPHGBIN 3. *jobname*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN

Table 116. DBCS translation table hierarchy (continued)

Program	Option	Customized translation tables	Default translation table
LPD Server	Kanji	1. *jobname*.LPD.TCPKJBIN 2. *hlq*.LPD.TCPKJBIN 3. *jobname*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
LPD Server	SChinese	1. *jobname*.LPD.TCPSCBIN 2. *hlq*.LPD.TCPSCBIN 3. *jobname*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
LPD Server	TChinese	1. *jobname*.LPD.TCPCHBIN 2. *hlq*.LPD.TCPCHBIN 3. *jobname*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN
SMTP Server	Hangeul	1. *jobname*.SMTP.TCPHGBIN 2. *hlq*.SMTP.TCPHGBIN 3. *jobname*.STANDARD.TCPHGBIN	*hlq*.STANDARD.TCPHGBIN
SMTP Server	Kanji	1. *jobname*.SMTP.TCPKJBIN 2. *hlq*.SMTP.TCPKJBIN 3. *jobname*.STANDARD.TCPKJBIN	*hlq*.STANDARD.TCPKJBIN
SMTP Server	SChinese	1. *jobname*.SMTP.TCPSCBIN 2. *hlq*.SMTP.TCPSCBIN 3. *jobname*.STANDARD.TCPSCBIN	*hlq*.STANDARD.TCPSCBIN
SMTP Server	TChinese	1. *jobname*.SMTP.TCPCHBIN 2. *hlq*.SMTP.TCPCHBIN 3. *jobname*.STANDARD.TCPCHBIN	*hlq*.STANDARD.TCPCHBIN

***:** See “Usage notes for the TRANSLATE option for the FTP client”

Notes:

1. *jobname* is the name specified either on the PROC or JOB statement.
2. *user_id* is the ID of the user who issued the command.
3. *data_set* is the name entered on the TRANSLATE parameter for the program. See the z/OS Communications Server: IP User's Guide and Commands for information about specifying the TRANSLATE parameter for the required program.

Usage notes for the TRANSLATE option for the FTP client

- To use the TRANSLATE option to load and use a customized DBCS translation table for the FTP client, an SBCS table data set must also exist for the *data_set_name* chosen with the TRANSLATE option.

  If the SBCS table data set does not exist, the FTP request fails even if a valid DBCS table data set using that name exists.
- ***ATTENTION***: Do not use the TRANSLATE option for the FTP client if the SBCS table you need for data transfer does not support standard encodings for the portable character set. Such a translation table can adversely affect the EBCDIC to ASCII conversion of commands sent over the control connection.

  For information about using FTP.DATA to specify different SBCS tables for control and data connections, see z/OS Communications Server: IP User's Guide and Commands.

If you require a local DBCS translation table, you must name it in correlation with the standard client search order. For example, if you had a custom Kanji table you could name it *user_id*.FTP.TCPKJBIN.

Telnet 3270 DBCS transform mode codefiles

The binary translation table code files used by Telnet 3270 DBCS transform mode do not use a search order hierarchy. If the DD statement is not specified, or the codefiles are not present, 3270 DBCS transform mode is disabled.

Restriction: The codefile members must reside in a data set pointed to by the TNDBCSXL DD statement in the TCPIPROC cataloged procedure.

Steps for customizing DBCS translation tables

You can find the DBCS translation tables in the installation libraries in both editable source and binary form. The Kanji, Hanguel, Traditional Chinese and Simplified Chinese DBCS editable source members reside in the SEZADBCX data set. The standard binary members reside in the STANDARD.TCPKJBIN for Kanji, the STANDARD.TCPHGBIN for Hangeul, the STANDARD.TCPCHBIN for Traditional Chinese, and the STANDARD.TCPSCBIN for Simplified Chinese. These data sets contain binary tables that are used by the FTP server, SMTP server, FTP client, LPR client, and LPD server programs. The binary codefiles used by Telnet 3270 DBCS transform mode reside in the SEZAXLD2 data set. The binary tables and codefiles can be created from the same editable source, using the CONVXLAT program.

Procedure

Perform the following steps to customize a DBCS translation table:

1. Make a copy of the editable source data set.
2. Modify the editable source as required.
3. Run the CONVXLAT program with the modified editable source as input.
4. Install the resulting customized binary table or codefiles in the DBCS translation table hierarchy for the required program.

Results

The editable source data sets contain two column pairs for each code page. The first column pair specifies double-byte EBCDIC-to-ASCII code point mappings for the indicated code page. The second column pair specifies double-byte ASCII-to-EBCDIC code point mappings for the indicated code page.

Existing code-point mappings can be changed by overwriting the existing hexadecimal code. Code points that are not defined in the target code page and are within the valid range for the code page are mapped to the default substitution character in the target code page. The default substitution characters are shown as (sub: xxxx) in the source tables in this topic.

The editable source format specifies EBCDIC-to-ASCII and ASCII-to-EBCDIC mappings separately. When adding or changing a code-point mapping, care should be taken to modify both mappings for the code point. If, for example, a new mapping is added for EBCDIC-to-ASCII only, the ASCII-to-EBCDIC mapping for that code-point is the default substitution character.

DBCS country or region translation tables

The translation table source members in Table 117are in the SEZADBCX data set. They are used by all applications that support DBCS. See "Using TSO CONVXLAT to convert translation tables to binary" for more information about using the TSO CONVXLAT command.

Requirement: If you modify these table members, you must convert them to binary format using CONVXLAT and store the modified binary data set in an appropriate data set within the DBCS translation table hierarchy.

Table 117. Translation table members for DBCS applications

Member name	Description	Code page
EZACHLAT (Taiwan DBCS)	TChinese Big5	0927<->0835 0947<->0835
EZAHGLAT (Korea DBCS)	Hangeul KSC5601	0926<->0834 0951<->0834
EZAKJLAT (Japan DBCS)	PC to host code page SJISKANJI 0941 at 1978	PC<->0300
EZAKJ941 (Japan DBCS)	PC to host code page SJISKANJI at 0941 at 1995 level	PC<->0300
EZASCLAT (People's Republic of China DBCS)	Schinese	1380<->0837

Syntax rules for DBCS translation tables

Observe the following rules for DBCS translation tables:

- Comments can be included in the editable data set, either on a separate line or at the end of a line. Comments must start with a semicolon (;).
- Code-point mappings in the data set are position dependent. The first non-comment line for the DBCS tables in the data set is used to establish the column position of the code point mappings, and must contain a conversion pair for each code page. Any conversion pairs on following lines must use the same column positions.
- It is permissible to leave blanks for code-point mappings after the first line in the DBCS area. For example, if a line contains only one conversion pair, the column position is used to determine which code page it refers to.
- The first column of each code page column pair (that is, the code index), must be in ascending numeric order. Any gaps in the ascending order is filled with the default substitution character in the binary table created by CONVXLAT.

Using TSO CONVXLAT to convert translation tables to binary

The TSO CONVXLAT command converts a table from editable text to binary. CONVXLAT can be used to convert both SBCS and DBCS table source data sets.

The syntax of the CONVXLAT command is:

```
►►──CONVXLAT──input_data_set──output_data_set──(──┬──────────────────────────┬──────►◄
                                                  ├─┬───────────┬──CODEFILE──┤
                                                  │ ├─KANJI─────┤            │
                                                  │ ├─HANGEUL───┤            │
                                                  │ ├─SCHINESE──┤            │
                                                  │ └─TCHINESE──┘            │
                                                  └─CODEFILE──(──member──)───┘
```

The parameters of the CONVXLAT command are:

input_data_set
: Specifies the source data set name to be converted. The data set name must be enclosed in quotation marks if fully qualified; otherwise the TSO user ID is appended as a prefix.

output_data_set
: Specifies the destination data set name created by the conversion. The data set name must be enclosed in quotation marks if fully qualified; otherwise the TSO user ID is appended as a prefix.

  Rule: If CODEFILE is also specified, then *output_data_set* must specify a previously allocated partitioned data set. Multiple codefile members are placed in the partitioned data set.

  The data set should be allocated using the following parameters:

```
Organization:         PO
Record format:        VB
Record length:        5124
Block size:           8800
1st extent blocks:    156
Secondary blocks:     10
```

KANJI
: Specifies that the tables being converted are the Japanese DBCS translation tables.

HANGEUL
: Specifies that the tables being converted are the Korean Standard DBCS translation tables.

SCHINESE
: Specifies that the table being converted is the Simplified Chinese DBCS translation table.

TCHINESE
: Specifies that the table being converted is the Traditional Chinese DBCS translation table.

CODEFILE
: Specifies that the selected table is converted to multiple codefiles for use in Telnet 3270 DBCS transform mode. The selected table must be DBCS translation table.

CODEFILE(*member***)**
: Specifies that the selected SBCS table is converted to two codefiles: ASCII_To_EBCDIC and EBCDIC_To_ASCII. The member names in the output PDS are *member*ATE and *member*ETA. The following names are possible member names:

J8E	JIS 8 Bit English
J8K	JIS 8 Bit Katakana
A8E	8 Bit English
A8K	8 Bit Katakana
SJDCE	DEC English SBCS
SJDCK	DEC Katakana SBCS
SJECE	Japanese EUC English SBCS

SJECK	Japanese EUC Katakana SBCS
SKSH	Korean KSC 5601 SBCS
SHAN	Hangeul SBCS
STCH	Traditional Chinese SBCS
SBG5	Big-5 SBCS
SSCH	Simplified Chinese SBCS

If no optional parameters are specified, the input data set is assumed to contain an SBCS translation table.

CONVXLAT examples

This topic contains CONVXLAT examples.

Running CONVXLAT in BATCH

The following examples are of running CONVXLAT in batch.

Run the CONVXLAT program directly in a job.

```
//S00100    EXEC PGM=CONVXLAT,
//          PARM='''TCP3AS.SEZATCPX(FRENCH)'' ''USER3.STANDARD.TCPXLBIN'''
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DUMMY,BLKSIZE=80
```

Run the CONVXLAT program by using TSO batch.

```
//TSOCNVXL JOB ,CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),USER=USER200,
//       PASSWORD=xxxxxx
//EXEC EXEC PGM=IKJEFT1B,TIME=(0,50),REGION=3096K
//*
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSTSIN DD *
convxlat 'TCP3AS.SEZATCPX(FRENCH)' +
        'user3.standard.tcpxlbin'
/*
```

SBCS binary table

The following example shows the creation of an SBCS binary table from user-provided editable text.

```
convxlat sbcs.source  standard.tcpxlbin
READY
```

French Telnet client SBC

The following example shows the creation of a French Telnet Client SBCS binary table for user ID user30 from the product provided editable text:

```
convxlat 'tcpip.v3r2.sezatelx(french)' 'user30.telnet.tcpxlbin'
READY
```

Korean KSC5601 SBCS and DBCS

The following example shows the creation of a Korean KSC5601 SBCS and DBCS binary table from the product-provided editable text. These tables can be used by FTP, LPR, LPD and SMTP.

```
convxlat 'tcpip.v3r2.sezatcpx(kor1088)' 'tcpip.v3r2.standard.tcpxlbin'
READY
convxlat 'tcpip.v3r2.sezadbcx(ezahglat)' 'tcpip.v3r2.standard.tcphgbin'(hangeul
READY
```

Big-5 and traditional Chinese

The following example shows the creation of Big-5 and Traditional Chinese SBCS and DBCS codefiles for use by the Telnet 3270 DBCS Transform facility:

```
convxlat 'tcpip.v3r2.sezatelx(TAI1114)'  'tcpip.v3r2.sezaxld2' (codefile(sbg5)
READY
convxlat 'tcpip.v3r2.sezatelx(TAI0904)'  'tcpip.v3r2.sezaxld2' (codefile(stch)
READY
convxlat 'tcpip.v3r2.sezadbcx(ezachlat)'   'tcpip.v3r2.sezaxld2' (tchinese codefile
EZA0652I Current code set is "TCHETA"
EZA0652I Current code set is "TCHATE"
EZA0652I Current code set is "BG5ETA"
EZA0652I Current code set is "BG5ATE"
READY
```

Japanese SBCS (CP 1041) and DBCS

The following example shows the creation of a Japanese SBCS (CP 1041) and DBCS binary table from the product provided editable text. These tables can be used by FTP, LPR, LPD and SMTP.

```
convxlat 'tcpip.v3r2.sezatcpx(JPNKANA)'  'tcpip.v3r2.standard.tcpxlbin'
READY
convxlat 'tcpip.v3r2.sezadbcx(ezakjlat)'  'tcpip.v3r2.standard.tcpkjbin'(kanji
READY
```

Japanese SBCS and DBCS codefiles

The following example shows the creation of Japanese SBCS and DBCS codefiles for use by the Telnet 3270 DBCS Transform facility:

```
convxlat 'tcpip.v3r2.sezatelx(J8E)'  'tcpip.v3r2.sezaxld2' (codefile(j8e)
READY
convxlat 'tcpip.v3r2.sezatelx(J8K)'  'tcpip.v3r2.sezaxld2' (codefile(j8k)
READY
convxlat 'tcpip.v3r2.sezatelx(A8E)'  'tcpip.v3r2.sezaxld2' (codefile(a8e)
READY
convxlat 'tcpip.v3r2.sezatelx(A8K)'  'tcpip.v3r2.sezaxld2' (codefile(a8k)
READY
convxlat 'tcpip.v3r2.sezatelx(SJECE)'   'tcpip.v3r2.sezaxld2' (codefile(sjece)
READY
convxlat 'tcpip.v3r2.sezatelx(SJECK)'   'tcpip.v3r2.sezaxld2' (codefile(sjeck)
READY
convxlat 'tcpip.v3r2.sezatelx(SJDCE)'   'tcpip.v3r2.sezaxld2' (codefile(sjdce)
READY
convxlat 'tcpip.v3r2.sezatelx(SJDCK)'   'tcpip.v3r2.sezaxld2' (codefile(sjdck)
READY
convxlat 'tcpip.v3r2.sezadbcx(ezakjlat)'   'tcpip.v3r2.sezaxld2' (kanji codefile
EZA0652I Current code set is "JIS78ETA"
EZA0652I Current code set is "JIS78ATE"
EZA0652I Current code set is "JIS83ETA"
EZA0652I Current code set is "JIS83ATE"
EZA0652I Current code set is "JEUCETA"
EZA0652I Current code set is "JEUCATE"
EZA0652I Current code set is "JDECETA"
EZA0652I Current code set is "JDECATE"
READY
```

Appendix B. LDAP definition files

This topic contains the policy definition files that define the policy schema characteristics to an LDAP server.

These files show, respectively, the definitions of the various attributes that can be used to define policies, and the definitions of the object classes that contain these attributes. See Chapter 21, “Policy Agent and policy applications,” on page 931 and z/OS Communications Server: IP Configuration Guide for guidance about the different types of policies and examples of their usage.

Restriction: Not all of the object classes and attributes shown in the definition files are supported on z/OS. The LDAP schema is a superset of policy object classes and attributes needed for several different platforms. Only those object classes and attributes shown in Chapter 21, “Policy Agent and policy applications,” on page 931 are supported.

PAGENTAT sample

```
#
#  pagent_at.conf
#
#  This file contains a set of LDAP directory attributes for the
#  Quality of Service (QOS) and Intrusion Detection System (IDS)
#  policy objects defined with the LDAP server.
#

# objectClass attribute is used to associate an object with a class (see
# object class definition file for detail).
#   This is a multi-valued attribute.
attribute       objectClass                         cis   objectClass               128 normal

# cn attribute specifies the common name of an object (e.g., a user friendly
# name and is often included in the object distinguished name).
#   This is a single-valued attribute.
attribute       cn                                  cis   cn                        128 normal

# ibm-policyKeywords attribute is used to provide a search filter for
# policy object retrieval.  This attribute applies to version 3
# policies.
#   This is a multi-valued attribute.
attribute       ibm-policyKeywords                  cis   policyKeywords            128 normal

# ibm-policyGroupName attribute specifies the user friendly name of a
# policyGroup object.
#   This is a single-valued attribute.
attribute       ibm-policyGroupName                 cis   policyGroupName            32 normal

# ibm-policyGroupKeywords attribute is used to provide a level of grouping
# together different policyGroup objects such that they can be searched
# and found together in one LDAP search (e.g., a way of scoping).
#   This is a multi-valued attribute.
attribute       ibm-policyGroupKeywords             cis   policyGroupKeywd          128 normal

# ibm-policyGroupsAuxContainedSet attribute provides an unordered set of
# distinguished name pointers to one or more policyGroup objects that
# are associated with the object to which this attribute has been
# appended.
#   This is a multi-valued attribute.  Its value is the distinguished
#   name of the referenced policyGroup object.
attribute       ibm-policyGroupsAuxContainedSet     dn    policyGroupsSet           256 normal
```

```
# ibm-policyRulesAuxContainedSet attribute provides an unordered set of
# distinguished name pointers to one or more policyRule objects that
# are contained within the object to which this attribute has been
# appended.
#   This is a multi-valued attribute.  Its value is the distinguished
#   name of the referenced policyRule object.
attribute       ibm-policyRulesAuxContainedSet         dn    policyRulesSet             256 normal

# ibm-policyGroupForLoadDistribution attribute provides a means to mark
# policy rules contained in a policy group as being intended for load
# distribution.  The S/390 implementation uses this attribute for
# policies to be interpreted on the Sysplex Distributor (SD)
# distributing stack.  NOTE: The S/390 implementation discards the policy
# group if a syntax error is detected on this attribute.  However, if
# any contained policy rules are retrieved outside the scope of the
# policy group, the default value of this attribute will be applied to
# them.  This attribute applies to version 2 policies.
#   This is a single-valued attribute.  Valid values are TRUE and FALSE.
#   The default is FALSE.
attribute       ibm-policyGroupForLoadDistribution     cis   policyGrpForLoadD           16 normal

attribute       description                            cis   description                256 normal

# ibm-policyRuleName attribute specifies the user friendly name of a
# policyRule object.
#   This is a single-valued attribute.
attribute       ibm-policyRuleName                     cis   policyRuleName              32 normal

# ibm-policyRuleEnabled attribute specifies an enumeration indicating
# whether a policy rule is administratively enabled, disabled, or
# enabled for debug mode.  Note that the S/390 implementation treats
# enabled for debug the same as enabled.
#   This is a single-valued attribute.  The defined values for this
#   attribute are 1 for enabled, 2 for disabled, and 3 for enabled for
#   debug mode.  Default is 1.
attribute       ibm-policyRuleEnabled                  cis   policyRuleEnable             1 normal

# ibm-policyRuleConditionListType attribute specifies whether the list of policy
# conditions associated with this policy rule is in Disjunctive Normal Form
# (DNF - ORed groups/sets of ANDed conditions) or Conjunctive Normal Form
# (CNF - ANDed groups/sets of ORed conditions).
#   This is a single-valued attribute.  The defined values for this
#   attribute are 1 for DNF, and 2 for CNF.  Default is 1.  Note that
#   this attribute is only valid for complex rules.
attribute       ibm-policyRuleConditionListType        cis   policyRuleCondLT             1 normal

# ibm-policyRuleConditionList attribute specifies an unordered list of strings
# of the form:
#     ibm-policyRuleConditionList:group-number:< +|- >:dn
# indicating a set of policy conditions that determine when the policy rule
# is applicable/fired.  The group-number specifies the group or set of
# the policy conditions, in which the referenced condition belongs.
# The < +|- > specifies if the condition is to be negated.  The dn is
# the distinguished name of the referenced condition.  This attribute
# applies to version 2 policies.
#   This is a multi-valued attribute.  Here is an example:
#      ibm-policyRuleConditionListType:1
#      ibm-policyRuleConditionList:1:+:C1
#      ibm-policyRuleConditionList:1:+:C2
#      ibm-policyRuleConditionList:2:+:C3
#      ibm-policyRuleConditionList:2:-:C4
#    This is equivalent to:    ( C1 AND C2 ) OR ( C3 AND (NOT C4) )
attribute       ibm-policyRuleConditionList            cis   policyRuleCondLi           256 normal

# ibm-policyRuleConditionListDN attribute specifies an unordered list of
# DN pointers indicating a set of policy conditions that determine when
# the policy rule is applicable/fired.  This attribute contains the
# distinguished name of the referenced condition.  This attribute
# applies to version 3 policies.
#   This is a multi-valued attribute.
attribute       ibm-policyRuleConditionListDN          dn    policyRuleCListD           256 normal

# ibm-policyRuleActionList attribute is an unordered list of strings of the form:
#    ibm-policyRuleActionList:n:dn
# it specifies an ordered set of policy actions to be performed if the
# overall associated policy conditions of the corresponding policy rule
# evaluates to TRUE.  The n value specifies the order of the actions
# to be executed.  A value of 0 means "don't care".  The dn is the
# distingushed name of the referenced action.  Note that the S/390
# implementation executes only one action that is found to be most
# appropriate (e.g., action with scope of DataTraffic or Both for
# non-RSVP IP traffic).  However, the actions are still ordered
# according to this attribute.  If there are more actions than can be
# executed, the first one in the ordered list will be selected and the
# remaining ones will be ignored.  This attribute applies to version
# 2 policies.
#   This is a multi-valued attribute.  Here is an example:
#      ibm-policyRuleActionList:1:DN-Action1
#      ibm-policyRuleActionList:2:DN-Action2
attribute       ibm-policyRuleActionList               cis   policyRuleActL             256 normal

# ibm-policyRuleActionListDN attribute is an unordered list of DN
# pointers to an ordered set of policy actions to be performed if the
# overall associated policy conditions of the corresponding policy rule
# evaluates to TRUE.  This attribute contains the distinguished name of
# the referenced action.  Note that the S/390 implementation executes
# only one action that is found to be most appropriate (e.g., action with
# scope of DataTraffic or Both for non-RSVP IP traffic).  However, the
# actions are still ordered according to the ibm-policyActionOrder
# attribute.  If there are more actions than can be executed, the first
# one in the ordered list will be selected and the remaining ones will be
# ignored.  This attribute applies to version 3 policies.
#   This is a multi-valued attribute.
attribute       ibm-policyRuleActionListDN             dn    policyRuleAListD           256 normal

# ibm-policyRuleValidityPeriodList attribute specifies the distinguished names
# of policyTimePeriodCondition objects that determine when the policy
# rule is scheduled to be active (inactive).
#   This is a multi-valued attribute.  Here is an example:
#      ibm-policyRuleValidityPeriodList:DN-timeperiod1
#      ibm-policyRuleValidityPeriodList:DN-timeperiod2
#   In this example, the policy rule will be active if the time is within
#   either the time specified in DN-timeperiod1 object or
#   DN-timeperiod2 object.
attribute       ibm-policyRuleValidityPeriodList       cis   policyRulePerL             256 normal

# ibm-policyRuleKeywords attribute is used to provide a level of grouping
# together different policyRule objects such that they can be initially
# searched and found together in one LDAP search (e.g., a way of scoping).
#   This is a multi-valued attribute.
attribute       ibm-policyRuleKeywords                 cis   policyRuleKeywd            128 normal

# ibm-policyRuleUsage attribute is used to provide guidelines on how the
# corresponding policy rule should be used.  S/390 will interpret this
# attribute but ignore its value.
#   This is a single-valued attribute.
attribute       ibm-policyRuleUsage                    cis   policyRuleUsage            128 normal

# ibm-policyRulePriority attribute specifies a non-negative integer for
# prioritizing a policy rule relative to other policy rules.  A larger
# value means higher priority.  Given two rules that are overlapped (they
# both cover some IP traffic), a rule with higher priority will be applied.
# The maximum supported value is 255.
#   This is a single-valued attribute.  Default value is zero.
attribute       ibm-policyRulePriority                 cis   policyRulePrio              32 normal

# ibm-policyRuleMandatory attribute is used as a flag to indicate that
# the evaluation of the policy conditions and execution of policy actions
# (when overall condition is evaluated to TRUE) is required.
#   This is a single-valued attribute.  Its value is either TRUE or FALSE.
#   Default is TRUE.  In S/390 implementation, it is always assumed to be
#   TRUE, therefore, this attribute is simply ignored.
attribute       ibm-policyRuleMandatory                cis   policyRuleMand              16 normal

# ibm-policyRuleSequencedActions attribute provides an integer enumeration to
# indicate how to interpret the action ordering indicated via the
# ibm-policyRuleActionList attribute (e.g., the n value), for version
# 2 policies, or the ibm-policyActionOrder attribute, for version 3
# policies.  The defined values for this attribute are: 1 (for
# mandatory), 2 (for recommended), and 3 (for don't care).  The default
# is 3.
#   This is a single-valued attribute.
attribute       ibm-policyRuleSequencedActions         cis   policyRuleSeqA               1 normal

# ibm-policyRoles attribute specifies a role or set of roles (known as a
# role-combination) that this policy plays.  Policy consumers (clients)
# can search for policy rules that contain one or more roles or
# role-combinations using this attribute.  Role-combinations are
# specified using the syntax:
#   role1&&role2...
# This attribute applies to version 3 policies.
#
#   This is a multi-valued attribute.
attribute       ibm-policyRoles                        cis   policyRoles                128 normal

# ibm-policyInstanceName attribute specifies the user friendly name of a
# ibm-policyInstance object.  This attribute applies to version 3
# policies.
#   This is a single-valued attribute.
attribute       ibm-policyInstanceName                 cis   policyInstName              32 normal

# ibm-policyConditionName attribute specifies the user friendly name of a
# ibm-policyCondition object.
#   This is a single-valued attribute.
attribute       ibm-policyConditionName                cis   policyCondName              32 normal

# ibm-policyConditionGroupNumber attribute specifies the group or set of
# the policy conditions to which a policy condition belongs.  These
# groups are used to form the DNF or CNF expression associated with a
# policy rule.  This attribute applies to version 3 policies.
#   This is a single-valued attribute.
attribute       ibm-policyConditionGroupNumber         cis   policyCondGrpNum            32 normal

# ibm-policyConditionNegated attribute specifies whether a policy
# condition is negated in the DNF or CNF expression associated with a
# policy rule.  A value of TRUE (meaning negated) or FALSE may be
# specified.  This attribute applies to version 3 policies.
#   This is a single-valued attribute.
attribute       ibm-policyConditionNegated             cis   policyCondNegate            32 normal

# ibm-policyConditionDN attribute specifies the distinguished name (DN)
# of a reusable policy condition.  This attribute applies to version 3
# policies.
#   This is a single-valued attribute.
attribute       ibm-policyConditionDN                  dn    policyCondDN               256 normal

# ibm-policyActionName attribute specifies the user friendly name
# of a ibm-policyAction object.  Up to 32 characters are supported (longer
# names are silently truncated).
#   This is a single-valued attribute.
attribute       ibm-policyActionName                   cis   policyActionName            32 normal

# ibm-policyActionOrder attribute specifies the relative order of the
# actions to be executed in the context of a policy rule.  A value of 0
# means "don't care".  Note that the S/390 implementation executes only
# one action that is found to be most appropriate (e.g., action with
# scope of DataTraffic or Both for non-RSVP IP traffic).  However, the
# actions are still ordered according to this attribute.  This attribute
# applies to version 3 policies.
#   This is a single-valued attribute.
attribute       ibm-policyActionOrder                  cis   policyActOrder              32 normal

# ibm-policyActionDN attribute specifies the distinguished name (DN)
# of a reusable policy action.  This attribute applies to version 3
# policies.
#   This is a single-valued attribute.
attribute       ibm-policyActionDN                     dn    policyActDN                256 normal

# ibm-sourceIPAddressRange attribute specifies the source addresses in IP
# packets to which the policy rule applies.  From a S/390 server's point
# of view, for inbound traffic, the source address in the IP packets will
# be the address of the client, whereas for outbound traffic, the source
# address will be one that is defined on the S/390 server (e.g., local
# subnet addresses including VIPA).  Either IPv4 or IPv6 addresses can
# be specified.  Here is the format of this attribute.
#    ibm-sourceIPAddressRange:n<-parameter according to 1 | 2 | 3 | 4 | 5 option>
#    ibm-sourceIPAddressRange:1    policy is applied to locally generated
#                              packets
#    ibm-sourceIPAddressRange:2-<IPv4Address>-<PrefixMaskLength>
#                              IPv4Address is in dotted decimal format.
#                              PrefixMaskLength is the number of unmasked
#                              leading bits.  An IP packet matches the condition
#                              if its source address unmasked bits are identical
#                              to the unmasked bits defined.
#    ibm-sourceIPAddressRange:3-<from-IPv4Address>[-<to-IPv4Address>]
#                              specifies IPv4Address range.
#                              to-IPv4Address has to be no less than from-IPv4Address.
#                              An IP packet matches the condition if its source
#                              address is within the range defined.
#    ibm-sourceIPAddressRange:4-<IPv6Address>-<PrefixMaskLength>
#                              IPv6Address is in colon-hex format.
#                              PrefixMaskLength is the number of unmasked
#                              leading bits.  An IP packet matches the condition
#                              if its source address unmasked bits are identical
#                              to the unmasked bits defined.
#    ibm-sourceIPAddressRange:5-<from-IPv6Address>[-<to-IPv6Address>]
#                              specifies IPv6Address range.
#                              to-IPv6Address has to be no less than from-IPv6Address.
#                              An IP packet matches the condition if its source
#                              address is within the range defined.
#   This is a single-valued attribute.
#    some examples:
#       ibm-sourceIPAddressRange:1
#       ibm-sourceIPAddressRange:2-9.87.65.43-24
#       ibm-sourceIPAddressRange:3-9.87.65.43-9.87.65.255
#       ibm-sourceIPAddressRange:5-1200::BA05
#          this last example contains only one address defined, no range.
attribute       ibm-sourceIPAddressRange               cis   sourceIPARange              64 normal

# ibm-destinationIPAddressRange attribute specifies the destination addresses in
# IP packets to which the policy rule applies.  From a S/390 server's point
# of view, for inbound traffic, the destination address in the IP packets will
# be the local address defined on the server, whereas for outbound traffic, the
# destination address will be the remote client's address.  Either IPv4 or
# IPv6 addresses can be specified.  Here is the format of this attribute:
#    ibm-destinationIPAddressRange:n<-parameter according to 1 | 2 | 3 | 4 | 5 option>
#    ibm-destinationIPAddressRange:1    policy is applied to locally destined
#                                   packets
#    ibm-destinationIPAddressRange:2-<IPv4Address>-<PrefixMaskLength>
#                              PrefixMaskLength is the number of unmasked
#                              leading bits.  An IP packet matches the condition
#                              if its destination address unmasked bits are
#                              identical to the unmasked bits defined.
#    ibm-destinationIPAddressRange:3-<from-IPv4Address>[-<to-IPv4Address>]
#                              specifies IPv4Address range.
#                              to-IPv4Address has to be no less than from-IPv4Address.
#                              An IP packet matches the condition if its
#                              destination address is within the range defined.
#    ibm-destinationIPAddressRange:4-<IPv6Address>-<PrefixMaskLength>
#                              PrefixMaskLength is the number of unmasked
#                              leading bits.  An IP packet matches the condition
#                              if its destination address unmasked bits are
#                              identical to the unmasked bits defined.
#    ibm-destinationIPAddressRange:5-<from-IPv6Address>[-<to-IPv6Address>]
#                              specifies IPv6Address range.
#                              to-IPv6Address has to be no less than from-IPv6Address.
#                              An IP packet matches the condition if its
#                              destination address is within the range defined.
#   This is a single-valued attribute.
#                    see ibm-sourceIPAddressRange for comments.
attribute       ibm-destinationIPAddressRange          cis   destIPARange                64 normal

# ibm-sourcePortRange attribute specifies the source application port number in the
# IP packets to which the policy rule applies.  From a S/390 server's point
# of view, for inbound traffic, the source port in an IP packet will
# be the remote client port, whereas for outbound traffic, the
# source port will be one of a local application in the server.
# Here is the format of this attribute:
#    ibm-sourcePortRange:<from-port>[:<to-port>]
#                    two integers that specify a port range.
#                    to-port has to be no less than from-port.
#                    An IP packet matches the condition if its
#                    source port is within the range defined.
#                    Note that port number can't exceed 16-bit field value.
#   This is a single-valued attribute.
#    some examples:
#       ibm-sourcePortRange:20:21
#       ibm-sourcePortRange:80
#          this last example contains only one port defined, no range.
attribute       ibm-sourcePortRange                    cis   sourcePortRange             32 normal

# ibm-destinationPortRange attribute specifies the destination application port number
# in the IP packets to which the policy rule applies.  From a S/390 server's
# point of view, for inbound traffic, the destination port in an IP packet will
# be the local application port in the server, whereas for outbound traffic, the
# destination port will be the remote client's port.
# Here is the format of this attribute:
#    ibm-destinationPortRange:<from-port>[:<to-port>]
#   This is a single-valued attribute.
#                    see ibm-sourcePortRange for comments.
attribute       ibm-destinationPortRange               cis   destPortRange               32 normal

# ibm-protocolNumberRange attribute specifies the protocol ID numbers in IP
# packets to which the policy rule applies.  The format of this attribute
# is as follows:
#    ibm-protocolNumberRange:<from-protocolID>[:<to-protocolID>]
#                    Two integers that specify a protocol ID range.
#                    to-protocolID has to be no less than from-protocolID.
#                    An IP packet matches the condition if its protocol
#                    ID value is within the range defined.
#                    Note that protocol number can't exceed 255 (8-bit field).
#   This is a single-valued attribute.
attribute       ibm-protocolNumberRange                cis   protoNumRange               32 normal

# ibm-applicationName attribute specifies the name of the application that
# is executing in the S/390 (e.g., also referred to as job name).  Application
# name is used when a predefined port number is not known for the application
# (e.g., applications that use dynamically assigned port numbers).  Note
# that in S/390, application names are converted to upper case for comparison
# with job names.  '*' can be used as a wildcard.  The specified name is
# limited to 8 characters (longer names are silently truncated).
# The format of this attribute is as follows:
#    ibm-applicationName:<name of the application/job in the system>
#   This is a single-valued attribute.
#    some examples:
#       ibm-applicationName:HTTPD
#       ibm-applicationName:FTPD*
attribute       ibm-applicationName                    cis   applName                     8 normal
```

```
# ibm-applicationData attribute is used for content-based policy classification.
# This means the policy allows policy condition to include application
# data to be included in the evaluation process.  It enables an application
# to assign different types of QoS treatments for different transactions
# (or streams of data) within a session.  In S/390, only web URI (Universal
# Resource Identifier) is supported as application data and only when the
# web application server activates Fast Response Cache Accelerator (FRCA)
# function.  This attribute is limited to 128 characters (longer data are
# silently truncated).  The format of this attribute is as follows:
#    ibm-applicationData:<a character string>
#   This is a single-valued attribute.
#    an example:
#       ibm-applicationData:/cat/purchase/info
attribute       ibm-applicationData                   ces   applData                   128 normal

# ibm-applicationPriority attribute is used for content-based policy
# classification.  It allows an application to assign different
# priorities for different transactions (or streams of data) within a
# session.  Valid values are as follows:
# 0 = Any application priority specified (default).
# 1 = EXPEDITED, 2 = HIGH, 3 = MEDIUM, 4 = LOW, 5 = BESTEFFORT.
# The format of this attribute is as follows:
#    ibm-applicationPriority:<an integer value>
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-applicationPriority:3
attribute       ibm-applicationPriority               cis   applPriority                 1 normal

# ibm-interface attribute is used for both ibm-policyRule and ibm-policyAction objects.
# For ibm-policyRule objects, it is used to limit the policy scope to specific
# inbound and outbound interfaces/subnets as IP packets traverse a network
# element (e.g., router).  If both inbound and outbound interface values
# are specified in an ibm-interface attribute, it means the corresponding
# policy is  to be applied to transit traffic that arrives on one interface
# and departs on another interface (e.g., traffic going through a router).
# From S/390 server's point of view, because our implementation of policy
# is as a host, a packet is destined to the server after it arrives on an
# inbound interface, whereas  an outbound packet originates from the server
# and is sent on an outbound interface.   As a result, if both inbound and
# outbound interface non-null values are specified together, the corresponding
# rule won't be mapped to any traffic since S/390 doesn't support policy as a
# routing node.  Either an IPv4 address or an interface name can be
# specified - the only way to specify IPv6 interfaces is by name.
# For ibm-policyAction objects, this attribute specifies a set of
# Sysplex Distributor routing interfaces (up to 32).  These routing interfaces
# are used by the SD routing component to choose among available servers
# in the S/390 sysplex.  An interface value of 0 can be specified to indicate
# that the SD router can use any available target server if none of the
# target servers identified with instances of this attribute are available.
# Only IPv4 addresses can be specified.
# The default is no policy control of Sysplex Distributor routing.
# The format of this attribute is as follows:
#    ibm-interface:1-[<In-Interface-IPv4Address>][-<Out_Interface-IPv4Address>]
#    ibm-interface:3-[<In-Interface-Name>][-<Out_Interface-Name>]
#       Type 1 is used for IPv4 addresses for ibm-PolicyRule and
#       ibm-PolicyAction.
#       Type 3 is used for IPv4 or IPv6 names for ibm-PolicyRule.
#       If either one of the inbound/outbound interfaces is not specified,
#       all inbound/outbound interfaces are assumed.  For ibm-PolicyAction
#       objects, only the outbound interface can be specified.
#   This is a multi-valued attribute.  However, it is treated as
#   single-valued for ibm-policyRule objects by the S/390 implementation.
#    some examples:
#       ibm-interace:1-9.87.65.43-9.87.60.1
#                    with this specification, the corresponding rule is
#                    to be applied when traffic enters interface 9.87.65.43
#                    and departs on interface 9.87.60.1.   As mentioned above,
#                    with S/390 implementation as a server, this
#                    corresponding rule WILL NOT be mapped.
#       ibm-interace:3-ETH1            no outbound specified
attribute       ibm-interface                         cis   interface                     64 normal

# ibm-serverDomainName attribute contains the name of the server
# specified in an HTTP request URL.
# The format of this attribute is as follows:
#    ibm-serverDomainName:<domain_name>
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
attribute       ibm-serverDomainName                  ces   destHDID                     128 normal

# ibm-userNameId attribute specifies the identity of the user that
# is requesting a service which is to be assigned a QoS level.
# The format of this attribute is as follows:
#    ibm-userNameId:<user_name>
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
attribute       ibm-userNameId                        ces   userName                      64 normal

# ibm-userQoSGroup attribute contains the QoS group that is used to
# classify a user that is requesting a service.
# The format of this attribute is as follows:
#    ibm-userQoSGroup:<group_name>
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
attribute       ibm-userQoSGroup                      ces   groupName                     64 normal

# ibm-IncomingTOS IncomingTOS attribute contains the value of the TOS
# (Type of Service) or DS (Differentiated Services) field to which incoming
# traffic will be classified along with other attributes (address, port
# numbers etc) for inbound traffic treatment.
# The format of this attribute is as follows:
#    ibm-IncomingTOS:<TOS_value>-<TOS_mask>
#               TOS_value is an 8 bit binary value, for example 01100000
#               TOS_mask is the number of significant bits in the mask,
#                 from 1 to 8.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
attribute       ibm-IncomingTOS                       cis   IncomingTOS                   16 normal

# ibm-idsConditionType attribute is used to specify the type of IDS
# conditions associated with a policy rule.  Valid values are ATTACK,
# for rules that specify attack conditions, TR, for Traffic Regulation
# rules, SCAN_GLOBAL, for the single rule that specifies global
# attributes for scan detection, or SCAN_EVENT, for individual scan
# detection rules.
# The format of this attribute is as follows:
#    ibm-idsConditionType:ATTACK | TR | SCAN_GLOBAL | SCAN_EVENT
# This attribute applies to version 3 policies.
#   This is a multi-valued attribute, although in most cases IDS rules
#   should specify only a single type.
#    an example:
#       ibm-idsConditionType:TR
attribute       ibm-idsConditionType                  cis   idsConditionType              32 normal

# ibm-idsAttackType attribute specifies the known types of intrusion
# attacks to be evaluated in conjunction with a policy rule.  Attacks
# are specified as follows:
#    MALFORMED_PACKET - specifies a rule for a number of specific
#      malformed packets that are detected on inbound traffic.
#    FLOOD - specifies a rule for flooding attacks.
#    OUTBOUND_RAW - specifies a rule to enforce restrictions on the use
#      of RAW sockets for outbound processing, to prevent this stack
#      from being used to attack other systems.  A list of restricted
#      IP protocols may also be specified in the rule's conditions.
#    ICMP_REDIRECT - specifies a rule for ICMP redirect detection.
#    PERPETUAL_ECHO - specifies a rule for preventing perpetual echos
#      over UDP ports.  A list of local UDP ports that always respond
#      to an input packet is also specified in the rule's conditions,
#      and a separate list of remote (network) UDP ports that always
#      respond is specified.  Use of this attack type is restricted to
#      using the CNF condition type, with exactly 3 CNF levels.  One
#      level provides the attack type of PERPETUAL_ECHO, one level
#      provides the local ports, and one level provides the remote
#      ports.  No other conditions may be specified in the rule.
#    IP_FRAGMENT - specifies a rule for detecting suspicious fragmented
#      packets.
#    RESTRICTED_IP_OPTIONS - specifies a rule to detect inbound IP
#      packets with IP options that are not allowed.  A list of
#      restricted IP options is also specified in the rule's conditions.
#    RESTRICTED_IP_PROTOCOL - specifies a rule to detect inbound IP
#      packets with IP protocols that are not allowed.  A list of
#      restricted IP protocols is also specified in the rule's
#      conditions.
# The format of this attribute is as follows:
#    ibm-idsAttackType:MALFORMED_PACKET | FLOOD | OUTBOUND_RAW |
#                      ICMP_REDIRECT | PERPETUAL_ECHO | IP_FRAGMENT |
#                      RESTRICTED_IP_OPTIONS | RESTRICTED_IP_PROTOCOL
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsAttackType:IP_FRAGMENT
attribute       ibm-idsAttackType                    cis   idsAttackType              32 normal

# ibm-idsIPOptionRange attribute specifies a list of restricted IP
# options for IDS attack rules.  This attribute is only valid when
# ibm-idsAttackType specifies RESTRICTED_IP_OPTIONS.
# The format of this attribute is as follows:
#    ibm-idsIPOptionRange:<from-option>[:<to-option>]
#                    two integers that specify an option range.
#                    to-option has to be no less than from-option.
#                    Note that option number can't exceed 255.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    some examples:
#       ibm-idsIPOptionRange:10:12
#       ibm-idsIPOptionRange:20
#          this last example contains only one option, no range.
attribute       ibm-idsIPOptionRange                 cis   idsIPOptionRange           32 normal

# ibm-idsLocalPortRange attribute specifies a list of local ports for
# IDS rules.
# The format of this attribute is as follows:
#    ibm-idsLocalPortRange:<from-port>[:<to-port>]
#                    two integers that specify a port range.
#                    to-port has to be no less than from-port.
#                    Note that port number can't exceed 65535.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    some examples:
#       ibm-idsLocalPortRange:8000:8009
#       ibm-idsLocalPortRange:12005
#          this last example contains only one port, no range.
attribute       ibm-idsLocalPortRange                cis   idsLclPortRange            32 normal

# ibm-idsRemotePortRange attribute specifies a list of remote ports for
# IDS rules.
# The format of this attribute is as follows:
#    ibm-idsRemotePortRange:<from-port>[:<to-port>]
#                    two integers that specify a port range.
#                    to-port has to be no less than from-port.
#                    Note that port number can't exceed 65535.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    some examples:
#       ibm-idsRemotePortRange:9000:9100
#       ibm-idsRemotePortRange:11100
#          this last example contains only one port, no range.
attribute       ibm-idsRemotePortRange               cis   idsRmtPortRange            32 normal
```

```
# ibm-idsProtocolRange attribute specifies a list of protocols for
# IDS rules.
# The format of this attribute is as follows:
#    ibm-idsProtocolRange:<from-protocol>[:<to-protocol>]
#                    two integers that specify a protocol range.
#                    to-protocol has to be no less than from-protocol.
#                    Note that protocol number can't exceed 255.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    some examples:
#       ibm-idsProtocolRange:100:105
#       ibm-idsProtocolRange:17
#          this last example contains only one protocol, no range.
attribute       ibm-idsProtocolRange                   cis   idsProtocolRange            32 normal

# ibm-idsLocalHostIPAddress attribute specifies a list of local IP
# addresses for IDS rules.
# The format of this attribute is as follows:
#    ibm-idsLocalHostIPAddress:n<-parameter according to 2 | 3 option>
#    ibm-idsLocalHostIPAddress:2-<IPv4Address>-<PrefixMaskLength>
#                              PrefixMaskLength is the number of unmasked
#                              leading bits.  An IP packet matches the condition
#                              if its local address unmasked bits are
#                              identical to the unmasked bits defined.
#    ibm-idsLocalHostIPAddress:3-<from-IPv4Address>[-<to-IPv4Address>]
#                              specifies an IPv4 address range.
#                              to-IPv4Address has to be no less than from-IPv4Address.
#                              An IP packet matches the condition if its
#                              local address is within the range defined.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    some examples:
#       ibm-idsLocalHostIPAddress:2-9.87.65.43-24
#       ibm-idsLocalHostIPAddress:3-9.87.65.43-9.87.65.255
#       ibm-idsLocalHostIPAddress:3-9.87.65.43
#          this last example contains only one address defined, no range.
attribute       ibm-idsLocalHostIPAddress              cis   idsLclIPAddress             64 normal

# ibm-idsRemoteHostIPAddress attribute specifies a list of remote IP
# addresses for IDS rules.
# The format of this attribute is as follows:
#    ibm-idsRemoteHostIPAddress:n<-parameter according to 2 | 3 option>
#    ibm-idsRemoteHostIPAddress:2-<IPv4Address>-<PrefixMaskLength>
#                              PrefixMaskLength is the number of unmasked
#                              leading bits.  An IP packet matches the condition
#                              if its remote address unmasked bits are
#                              identical to the unmasked bits defined.
#    ibm-idsRemoteHostIPAddress:3-<from-IPv4Address>[-<to-IPv4Address>]
#                              specifies an IPv4 address range.
#                              to-IPv4Address has to be no less than from-IPv4Address.
#                              An IP packet matches the condition if its
#                              remote address is within the range defined.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    some examples:
#       ibm-idsRemoteHostIPAddress:2-129.10.11.0-23
#       ibm-idsRemoteHostIPAddress:3-9.10.11.0-9.10.11.255
#       ibm-idsRemoteHostIPAddress:3-211.0.42.1
#          this last example contains only one address defined, no range.
attribute       ibm-idsRemoteHostIPAddress             cis   idsRmtIPAddress             64 normal

# ibm-ptpConditionTime attribute specifies the range of calendar dates on which
# the corresponding policy rule is valid.  The format of this attribute is as
# follows:
#    ibm-ptpConditionTime:yyyymmddhhmmss:yyyymmddhhmmss
#    where yyyy is year, mm is month, dd is date, hh is hour, mm is minute and
#    ss is second.  Seconds are rounded to the nearest minute.  Default is
#    always.  Out of bounds values are forced to be correct (for instance
#    month 13 becomes January of the following year).  Dates before the
#    start of the Posix epoch (Jan/01/1970 00:00:00 UTC) are not valid.  The
#    time is kept in the format of seconds since the epoch - this value
#    wraps early in the year 2038, so times after that are not valid.
#   This is a single-valued attribute.
#    an example:
#       ibm-ptpConditionTime:19990101080000:20021231170000
#         (translates to:  from Jan/01/1999 8AM to Dec/31/2002 5PM)
attribute       ibm-ptpConditionTime                   cis   ptpConditionTime            64 normal

# ibm-ptpConditionMonthOfYearMask attribute specifies a mask identifying the months
# of the year in which the corresponding policy rule is valid.  The format
# of this attribute is as follows:
#    ibm-ptpConditionMonthOfYearMask:<a string of 12 '0's and '1's>
#   This is a single-valued attribute.
#    an example:
#       ibm-ptpConditionMonthOfYearMask:111000000000
#                                       (Jan, Feb, and March)
attribute       ibm-ptpConditionMonthOfYearMask        cis   ptpCondMonMask              12 normal

# ibm-ptpConditionDayOfMonthMask attribute specifies a mask identifying the days
# of the month on which the corresponding policy rule is valid.  The format
# of this attribute is as follows:
#    ibm-ptpConditionDayOfMonthMask:<a string of 31 or 62 '0's and '1's>
#    The first 31 bits identify the days of the month in the forward
#    direction (from the first day to the last day).  The last 31 bits,
#    which are optional, identify the days of the month in the reverse
#    direction (from the last day to the first day).  For example,
#    the 32nd bit represents the 31st day in January, but represents
#    the 29th day in February in a leap year.
#                                   Default is all month long.
#   This is a single-valued attribute.
#    an example:
#       ibm-ptpConditionDayOfMonthMask:1111111111111110000000000000000
#                                      (first 15 days of the month)
attribute       ibm-ptpConditionDayOfMonthMask         cis   ptpCondDayMonM              64 normal

# ibm-ptpConditionDayOfWeekMask attribute specifies a mask identifying the days
# of the week on which the corresponding policy rule is valid.  The format
# of this attribute is as follows:
#    ibm-ptpConditionDayOfWeekMask:<a string of 7 '0's and '1's beginning with Sunday>
#                                  Default is all week long.
#   This is a single-valued attribute.
#    an example:
#       ibm-ptpConditionDayOfWeekMask:0111110
#                                     (weekdays)
attribute       ibm-ptpConditionDayOfWeekMask          cis   ptpCondDayWeekM              8 normal

# ibm-ptpConditionTimeOfDayMask attribute specifies a range of times in a day
# during which the corresponding policy rule is valid.  The format
# of this attribute is as follows:
#    ibm-ptpConditionTimeOfDayMask:hhmmss:hhmmss
#       The second time identifies later time than the first.  When it is
#       smaller the time range spans midnight.  Seconds are rounded to
#       the nearest minute.  Default is 24 hours.
#   This is a single-valued attribute.
#    some examples:
#       ibm-ptpConditionTimeOfDayMask:080000:170000
#                                     (8AM to 5PM)
#       ibm-ptpConditionTimeOfDayMask:170000:080000
#                                     (5PM to 8AM the next day)
attribute       ibm-ptpConditionTimeOfDayMask          cis   ptpCondTimeDayM             16 normal

# ibm-ptpConditionTimeZone attribute specifies the time zone for which to
# apply the time specified in the ibm-policyTimePeriodCondition.  The format
# of this attribute is as follows:
#    ibm-ptpConditionTimeZone:< either Z or <'+' | '-'><hh[mm]> >
#                         Z indicates UTC
#                         '+' or '-' represents east or west of UTC.
#                         +/-0 is the same as UTC.
#                         hhmm is the hour and minutes from UTC (up to
#                         +/-1359).  Minutes are optional.  Default is
#                          local time.
#   This is a single-valued attribute.
#    an example:
#       ibm-ptpConditionTimeZone:+0400
#                                 (4 hours east of UTC)
attribute       ibm-ptpConditionTimeZone              cis   ptpCondTimeZone              8 normal

# ibm-ptpConditionLocalOrUtcTime attribute specifies whether the time
# zone to be applied to the time specified in the ibm-policyTimePeriodCondition
# is in local time or UTC time.  This attribute applies to version 3
# policies.
#   This is a single-valued attribute.  The defined values for this
#   attribute are 1 for local time and 2 for UTC time.  The default is 1.
attribute       ibm-ptpConditionLocalOrUtcTime        cis   ptpCondLocalOrUtc            8 normal

# ibm-PolicyScope attribute identifies the type of QoS service that the
# corresponding policy action specifies.  It can either be DataTraffic
# (aka DiffServ for Differentiated Services) or RSVP (for Resource reSerVation
# Protocol) or Both.  Based on the policy scope, a set of corresponding
# parameters can be applied for the traffic that is mapped to the policy
# action.
# The format of this attribute is as follows:
#    ibm-PolicyScope:<DataTraffic | RSVP | Both>
#             Default is Both.
#   This is a single-valued attribute.
#    an example:
#       ibm-PolicyScope:DataTraffic
attribute       ibm-PolicyScope                       cis   PolicyScope                 16 normal

# ibm-Permission attribute specifies whether or not to accept or deny traffic
# that is mapped to the corresponding policy action.
# The format of this attribute is as follows:
#    ibm-Permission:<Blocked | Allowed>
#             Default is Allowed.
#   This is a single-valued attribute.
attribute       ibm-Permission                        cis   Permission                   8 normal

# ibm-MaxRate attribute specifies the maximum TCP throughput for a connection
# that is mapped to the corresponding policy action.   It is used to control
# the upper limit of the TCP congestion window with respect to the roundtrip
# time.  The format of this attribute is as follows:
#    ibm-MaxRate:<an integer number in Kbps>
#             Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-MaxRate                           cis   MaxRate                     32 normal

# ibm-MinRate attribute specifies the minimum TCP throughput for a connection
# that is mapped to the corresponding policy action.   It is used to control
# the lower limit of the TCP congestion window with respect to the roundtrip
# time.  The format of this attribute is as follows:
#    ibm-MinRate:<an integer number in Kbps>
#             Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-MinRate                           cis   MinRate                     32 normal

# ibm-MaxDelay attribute specifies the maximum TCP roundtrip delay for a connection
# that is mapped to the corresponding policy action.   It is used mainly for
# policy performance monitor and/or profiling (see SLAPM MIB).
# The format of this attribute is as follows:
#    ibm-MaxDelay:<an integer number in milliseconds>
#             Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-MaxDelay                          cis    MaxDelay                   32 normal

# ibm-OutgoingTOS attribute specifies the IP TOS byte (Type Of Service, aka
# Differentiated Services - DS byte) value to be set for outgoing IP packets
# that are mapped to the corresponding policy action from S/390.  This TOS/DS
# byte also determines the priority queue in which to place packets for S/390
# QDIO devices.
# The format of this attribute is as follows:
```

```
#    ibm-OutgoingTOS:<a string of 8 '0' and '1'>
#             Default is 0.
#   This is a single-valued attribute.
#    an example:
#       ibm-OutgoingTOS:11000000
attribute       ibm-OutgoingTOS                      cis   OutgoingTOS                  8 normal

# ibm-MaxConnections attribute specifies the maximum number of TCP connections
# that are allowed within the policy action that contains this attribute.
# When this number is reached, additional TCP connections whose traffic is
# mapped to a policy rule which references the corresponding action are
# denied.
# The format of this attribute is as follows:
#    ibm-MaxConnections:<an integer number>
#             Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-MaxConnections                   cis   MaxConnections              32 normal

# ibm-DiffServInProfileRate attribute specifies the mean rate (token generating
# rate) of a token bucket traffic conditioner that enforces the rate of
# traffic that is mapped to the corresponding policy action by a policy rule.
# If the traffic exceeds this rate, it will be considered as out-of-profile
# and therefore will be treated with the action specified in
# ibm-DiffServExcessTrafficTreatment attribute.  If this value is non-zero,
# but ibm-DiffServInProfileTokenBucket is zero, then no token bucket traffic
# enforcement is performed.
# The format of this attribute is as follows:
#    ibm-DiffServInProfileRate:<an integer number in Kbps>
#             Default is no token bucket enforcement of traffic.
#   This is a single-valued attribute.
attribute       ibm-DiffServInProfileRate            cis   DSInProfRate                32 normal

# ibm-DiffServInProfilePeakRate attribute specifies the peak rate of a token bucket
# traffic conditioner that enforces the peak rate of traffic that is mapped to
# the corresponding policy action by a policy rule.  If the traffic exceeds
# this rate, it will be considered as out-of-profile and therefore will
# be treated with the ibm-DiffServExcessTrafficTreatment attribute.  If this
# value is non-zero, but ibm-DiffServInProfileMaxPacketSize or
# ibm-DiffServInProfileRate is zero, then no token bucket peak rate enforcement
# is performed.  If this value is less than ibm-DiffServInProfileRate, then
# no token bucket traffic or peak rate enforcement is performed.
# The format of this attribute is as follows:
#    ibm-DiffServInProfilePeakRate:<an integer number in Kbps>
#             Default is no token bucket enforcement of peak rate.
#   This is a single-valued attribute.
attribute       ibm-DiffServInProfilePeakRate        cis   DSInProfPeakRt              32 normal

# ibm-DiffServInProfileTokenBucket attribute specifies the maximum burst size of
# a token bucket traffic conditioner that enforces the burst of traffic
# that is mapped to the corresponding policy action by a policy rule.  It is
# used together with the mean rate in generating tokens consumed by outgoing
# traffic.
# The format of this attribute is as follows:
#    ibm-DiffServInProfileTokenBucket:<an integer number in Kb>
#             Default is 100.
#   This is a single-valued attribute.
attribute       ibm-DiffServInProfileTokenBucket     cis  DSInProfTB                   32 normal

# ibm-DiffServInProfileMaxPacketSize attribute specifies the maximum size of an
# IP packet being enforced by a token bucket traffic conditioner.
# Note that due to blocking in S/390, multiple packets tend to be sent
# back to back and if maximum packet size is just big enough for one packet,
# violation of the peak rate (peak rate enforcement is based on the size of
# each individual packet) will result and violated packets will be sent with
# different TOS value or dropped as a consequence.  To accommodate this
# blocking, the value of this attribute should be set in multiples of the
# maximum packet size (e.g., equal to the token bucket size).
# The format of this attribute is as follows:
#    ibm-DiffServInProfileMaxPacketSize:<an integer number in Kb>
#             Default is 100.
#   This is a single-valued attribute.
attribute       ibm-DiffServInProfileMaxPacketSize     cis   DSInProfMPS                 32 normal

# ibm-DiffServOutProfileTransmittedTOSByte attribute specifies the TOS value to
# be used for out-of-profile traffic if the excess treatment specified is
# to send them as best effort.
# The format of this attribute is as follows:
#    ibm-DiffServOutProfileTransmittedTOSByte:<a string of 8 '0' and '1'>
#            Default is 0.
#   This is a single-valued attribute.
attribute       ibm-DiffServOutProfileTransmittedTOSByte  cis  DSOutProfTosB              8 normal

# ibm-DiffServExcessTrafficTreatment attribute specifies how a token bucket
# traffic conditioner should treat out-of-profile traffic.  Two options
# can be specified, either Drop or BestEffort.  If treatment is to send
# BestEffort, a different TOS value, if specified, will be used.  If
# treatment is to Drop, depending on whether the traffic is UDP or TCP
# different mechanisms will be used to handle Drop treatment:
#   For UDP, traffic will actually be dropped.
#   For TCP, Drop treatment is simulated in that TCP congestion window is
#     cut (just as the case when a packet is dropped) immediately but the
#     violated packet will be sent.  This is to avoid overhead associated
#     with retransmission processing and also to reduce the traffic
#     generated immediately without having to wait for a roundtrip time
#     (i.e., standard TCP lost detection delay).  Also, TCP connections
#     that are mapped to the same policy (i.e., aggregation) will share
#     the throughput equally among them.
# The format of this attribute is as follows:
#    ibm-DiffServExcessTrafficTreatment:<Drop | BestEffort>
#            Default is BestEffort.
#   This is a single-valued attribute.
attribute       ibm-DiffServExcessTrafficTreatment     cis   DSExcessTreat               16 normal

# ibm-FlowServiceType attribute specifies the reservation type
# that can be requested by an RSVP flow, either ControlledLoad
# or Guaranteed.  Guaranteed service is considered to be greater than
# ControlledLoad.  Use this attribute to limit the service type requested
# from RSVP applications.
#    ibm-FlowServiceType:<ControlledLoad | Guaranteed>
#            Default is ControlledLoad
#   This is a single-valued attribute.
attribute       ibm-FlowServiceType                    cis   FlowServiceType             32 normal

# ibm-MaxRatePerFlow attribute specifies the maximum rate
# that can be requested by an RSVP flow that is mapped to a policy rule
# which references the corresponding policy action containing this attribute.
# The format of this attribute is as follows:
#    ibm-MaxRatePerFlow:<an integer number in Kbps>
#            Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-MaxRatePerFlow                     cis   MaxRatePerFlow              32 normal

# ibm-MaxTokenBucketPerFlow attribute specifies the maximum token bucket size
# that can be requested by an RSVP flow that is mapped to a policy rule
# which references the corresponding policy action containing this attribute.
# The format of this attribute is as follows:
#    ibm-MaxTokenBucketPerFlow:<an integer number in Kbits>
#            Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-MaxTokenBucketPerFlow              cis   MaxTBPerFlow                32 normal

# ibm-MaxFlows attribute specifies the maximum number of RSVP flows that are
# allowed within the policy action that contains this attribute.  When
# this number is reached, additional RSVP flow requests that are mapped
# to a policy rule which references the corresponding action are denied.
# The format of this attribute is as follows:
#    ibm-MaxFlows:<an integer number>
#            Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-MaxFlows                           cis   MaxFlows                    32 normal
```

```
# ibm-signalClient attribute specifies additional QoS function to the
# traditional sockets functions calls that will enable RSVP processing for
# a TCP or UDP connection.
# The format of this attribute is as follows:
#    ibm-signalClient:< 0 = No Signaling | 1 = Signaling >
#             Default is 1 (Signaling).
#   This is a single-valued attribute.
attribute       ibm-SignalClient                      cis   signalClient                32 normal

# ibm-inboundScope attribute identifies the type of inbound QoS service
# that the corresponding policy action specifies.  It can be either
# Application (for a named application) or Connection (for general TCP
# connections).  Based on the inbound scope, a set of corresponding
# parameters can be applied for the traffic that is mapped to the policy
# action. This attribute is extendable to other applications.
# The format of this attribute is as follows:
#    ibm-inboundScope:< Application | Connection >
#             Default is Connection.
#   This is a single-valued attribute.
attribute       ibm-inboundScope                      cis   inboundScope                16 normal

# ibm-averageConnectionRate attribute specifies the average number of new
# requests (connections) admitted per second.  If either the
# ibm-averageConnectionRate or ibm-connectionBurstSize is not in profile
# then the inbound connection will be <Drop>.
# The format of this attribute is as follows:
#    ibm-averageConnectionRate:<an integer number>
#             Default is 100.
#   This is a single-valued attribute.
attribute       ibm-averageConnectionRate             cis   averageConnRate             32 normal

# ibm-PeakConnectionRate attribute specifies the peak rate of a token bucket
# traffic conditioner that enforces the peak rate of traffic that is mapped
# to the corresponding inbound policy action by a policy rule.  If the number
# of connections exceeds this rate, it will be considered as out-of-profile
# and therefore will be treated is if the ibm-DiffServExcessTrafficTreatment
# attribute was set to <Drop>.
# The format of this attribute is as follows:
#    ibm-PeakConnectionRate:<an integer number>
#             Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-PeakConnectionRate                cis   PeakConnRate                32 normal

# ibm-connectionBurstSize attribute specifies the maximum number of new
# requests (connections) accepted  concurrently.  If either the
# ibm-averageConnectionRate or ibm-connectionBurstSize is not in profile
# then the inbound connection will be <Drop>.
# The format of this attribute is as follows:
#    ibm-connectionBurstSize:<an integer number>
#             Default is 5.
#   This is a single-valued attribute.
attribute       ibm-connectionBurstSize               cis   connBurstSize               32 normal

# ibm-averageApplicationRequestRate attribute specifies the average number
# of new application requests admitted per second.  If either the
# ibm-averageApplicationRequestRate or ibm-applicationRequestBurstSize is
# not in profile then the inbound request will be <Drop>.
# The format of this attribute is as follows:
#    ibm-averageApplicationRequestRate:<an integer number>
#             Default is 100.
#   This is a single-valued attribute.
attribute       ibm-averageApplicationRequestRate     cis   averageApplReqRat           32 normal

# ibm-applicationRequestPeakRate attribute specifies the peak rate of a
# token bucket traffic conditioner that enforces the peak rate of traffic
# that is mapped to the corresponding inbound policy action by a policy
# rule.  If the number of application requests exceeds this rate, it will be
# considered out-of-profile and the inbound application request will be
# <Drop>.
# The format of this attribute is as follows:
#    ibm-applicationRequestPeakRate:<an integer number>
#            Default is no limit.
#   This is a single-valued attribute.
attribute       ibm-applicationRequestPeakRate        cis   applRequestPeakRa           32 normal

# ibm-applicationRequestBurstSize attribute specifies the maximum number of
# new application requests accepted  concurrently.  If either the
# ibm-averageApplicationRequestRate or ibm-applicationRequestBurstSize is
# not in profile then the inbound request will be <Drop>.
# The format of this attribute is as follows:
#    ibm-applicationRequestBurstSize:<an integer number>
#            Default is 5.
#   This is a single-valued attribute.
attribute       ibm-applicationRequestBurstSize       cis   applRequestBurstS           32 normal

# ibm-prioritizedQueue attribute specifies the order the queue of the server
# processes incoming connections.  If the incoming packet is within the
# profiles limits then each connection will be served by one of 3 priorties.
# The format of this attribute is as follows:
#    ibm-prioritizedQueue:<an integer number 1 = High | 2 = Medium | 3 = Low>
#            Default is 2 - Medium.
#   This is a single-valued attribute.
attribute       ibm-prioritizedQueue                  cis   prioritizedQueue            32 normal

# ibm-idsActionType attribute is used to specify the type of IDS
# actions associated with a policy rule.  Valid values are ATTACK,
# for rules that specify attack actions, TR, for Traffic Regulation
# actions, SCAN_GLOBAL, for the single action that specifies global
# attributes for scan detection, or SCAN_EVENT, for individual scan
# detection actions.
# The format of this attribute is as follows:
#    ibm-idsActionType:ATTACK | TR | SCAN_GLOBAL | SCAN_EVENT
# This attribute applies to version 3 policies.
#   This is a multi-valued attribute.
#    an example:
#       ibm-idsActionType:SCAN_EVENT
attribute       ibm-idsActionType                     cis   idsActionType               32 normal

# ibm-idsNotification attribute specifies the types of notification to
# be provided for the events mapped by the corresponding IDS rule.
# Valid values are NONE, for no notification, SYSLOG, to log to the
# syslog daemon (syslogd), SYSLOGDETAIL, to log more detailed information
# to syslogd, or CONSOLE, to log to the system console.
# The format of this attribute is as follows:
#    ibm-idsNotification:NONE | SYSLOG | SYSLOGDETAIL | CONSOLE
#            The default is NONE.
# This attribute applies to version 3 policies.
#   This is a multi-valued attribute, but NONE can't be specified with
#   any other values.
#    an example:
#       ibm-idsNotification:CONSOLE
attribute       ibm-idsNotification                   cis   idsNotification             32 normal

# ibm-idsStatInterval attribute specifies the interval length in minutes
# for collecting IDS statistics.
# The format of this attribute is as follows:
#    ibm-idsStatInterval:<an integer number>
#            The default is 60.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsStatInterval:600
attribute       ibm-idsStatInterval                   cis   idsStatInterval             32 normal

# ibm-idsLoggingLevel attribute specifies the syslogd logging level for
# logging IDS information.  Valid values are 0 through 7.
# The format of this attribute is as follows:
#    ibm-idsStatInterval:<an integer number>
#            The default is 0.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsLoggingLevel:1
attribute       ibm-idsLoggingLevel                    cis   idsLoggingLevel             32 normal

# ibm-idsTypeActions attribute specifies the type of actions to be taken
# for IDS events.  Valid values are STATISTICS, for collecting statistics
# information only, EXCEPTSTATS, for collecting exception statistics
# only, LOG, to log IDS information according to the ibm-idsNotification
# attribute, or LIMIT, to enforce IDS Traffic Regulation limits and to
# cause detected attack packets to be dropped.
# The format of this attribute is as follows:
#    ibm-idsTypeActions:STATISTICS | EXCEPTSTATS | LOG | LIMIT
# This attribute applies to version 3 policies.
#   This is a multi-valued attribute.
#    an example:
#       ibm-idsTypeActions:LOG
attribute       ibm-idsTypeActions                     cis   idsTypeActions              32 normal

# ibm-idsTraceData attribute specifies the amount of information written
# to the IDS event trace.  Valid values are NONE, for no tracing, HEADER
# for tracing the IP and transport headers in packets, FULL, for tracing
# entire packets, or RECORDSIZE, for tracing the amount of data specified
# with the ibm-idsTraceRecordSize attribute (this amount of data
# includes the IP and transport headers).
# The format of this attribute is as follows:
#    ibm-idsTraceData:NONE | HEADER | FULL | RECORDSIZE
#             The default is HEADER.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsTraceData:RECORDSIZE
attribute       ibm-idsTraceData                       cis   idsTraceData                32 normal

# ibm-idsTraceRecordSize attribute specifies the amount of packet data
# to trace, when ibm-idsTraceData:RECORDSIZE is specified.
# The format of this attribute is as follows:
#    ibm-idsTraceRecordSize:<an integer number>
#             The default is 100.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsTraceRecordSize:50
attribute       ibm-idsTraceRecordSize                 cis   idsTraceRecordSz            32 normal

# ibm-idsMaxEventMessage attribute specifies the maximum number of event
# messages to be displayed on the console during a 5 minute period for
# an IDS attack type (e.g. MALFORMED_PACKET).
# The format of this attribute is as follows:
#    ibm-idsMaxEventMessage:<an integer number>
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsMaxEventMessage:5
attribute       ibm-idsMaxEventMessage                 cis   idsMaxEventMsg              32 normal

# ibm-idsIfcFloodPercentage attribute specifies the percentage of
# discarded packets for an interface above which an interface flood
# attack is recognized.  The minimum value is 5%.  The default is 10%.
# The format of this attribute is as follows:
#    ibm-idsIfcFloodPercentage:<an integer number>
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsIfcFloodPercentage:20
attribute       ibm-idsIfcFloodPercentage              cis   idsIfcFloodPrcnt            32 normal

# ibm-idsIfcFloodMinDiscard attribute specifies the minimum number of
# discarded packets for an interface that must occur within a 1 minute
# period in order to be recognized as an interface flood attack.  The
# minimum value is 100.  The default is 1000.
# The format of this attribute is as follows:
#    ibm-idsIfcFloodMinDiscard:<an integer number>
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsIfcFloodMinDiscard:500
attribute       ibm-idsIfcFloodMinDiscard             cis   idsIfcFloodMinDs            32 normal

# ibm-idsTRtcpTotalConnections attribute specifies the size of the total
# connection pool for IDS TCP Traffic Regulation functions.  The maximum
# value is 65535.
# The format of this attribute is as follows:
#    ibm-idsTRtcpTotalConnections:<an integer number>
#             The default is 65535.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsTRtcpTotalConnections:1000
attribute       ibm-idsTRtcpTotalConnections          cis   idsTRtcpTotConn             32 normal

# ibm-idsTRtcpPercentage attribute specifies the percentage of the total
# connections allowed with the ibm-idsTRtcpTotalConnections attribute
# that can be used by a single host.  The range is 0 - 100%.
# The format of this attribute is as follows:
#    ibm-idsTRtcpPercentage:<an integer number>
#             The default is 100.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsTRtcpPercentage:50
attribute       ibm-idsTRtcpPercentage                cis   idsTRtcpPercent             32 normal

# ibm-idsTRtcpLimitScope attribute specifies the scope of TCP traffic
# regulation.  Valid values are PORT, meaning that traffic regulation
# parameters apply to the aggregate of all sockets bound to the target
# port, or PORT_INSTANCE, meaning that traffic regulation parameters
# apply to each socket bound to the target port individually.
# The format of this attribute is as follows:
#    ibm-idsTRtcpLimitScope:PORT | PORT_INSTANCE
#             The default is PORT_INSTANCE.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsTRtcpLimitScope:PORT
attribute       ibm-idsTRtcpLimitScope                cis   idsTRtcpLmtScope            32 normal

# ibm-idsTRudpQueueSize attribute specifies the size of the port backlog
# queue.  This attribute is used to select one of a number of abstract
# queue sizes that map to internally defined limits.  Valid values are
# VERY_LONG, LONG, SHORT, VERY_SHORT.
# The format of this attribute is as follows:
#    ibm-idsTRudpQueueSize:VERY_LONG | LONG | SHORT | VERY_SHORT
#             The default is VERY_LONG.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsTRudpQueueSize:SHORT
attribute       ibm-idsTRudpQueueSize                 cis   idsTRudpQueueSize           32 normal

# ibm-idsFSInterval attribute specifies the interval in minutes for
# monitoring for fast scanning attacks.  The maximum value is 1440.
# Only one policy rule in the set of rules for a given stack can specify
# this global scan value.
# The format of this attribute is as follows:
#    ibm-idsFSInterval:<an integer number>
#             The default is 1.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsFSInterval:10
attribute       ibm-idsFSInterval                     cis   idsFSInterval              32 normal

# ibm-idsFSThreshold attribute specifies the fast scanning threshold.
# A fast scan attack is detected if more events from a single source
# are detected than specified within the interval defined with the
# ibm-idsFSInterval attribute.  The maximum value is 64.  Only one
# policy rule in the set of rules for a given stack can specify this
# global scan value.
# The format of this attribute is as follows:
#    ibm-idsFSThreshold:<an integer number>
#             The default is 5.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsFSThreshold:9
attribute       ibm-idsFSThreshold                    cis   idsFSThreshold             32 normal

# ibm-idsSSInterval attribute specifies the interval in minutes for
# monitoring for slow scanning attacks.  The maximum value is 1440.
# The value specified must be greater than the value specified for the
# ibm-idsFSInterval attribute.  However, a value of 0 can be specified
# to indicate that no slow scan processing should be performed.  Only
# one policy rule in the set of rules for a given stack can specify
# this global scan value.
# The format of this attribute is as follows:
#    ibm-idsSSInterval:<an integer number>
#             The default is 120.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsSSInterval:300
attribute       ibm-idsSSInterval                     cis   idsSSInterval              32 normal

# ibm-idsSSThreshold attribute specifies the slow scanning threshold.
# A slow scan attack is detected if more events from a single source
# are detected than specified within the interval defined with the
# ibm-idsSSInterval attribute.  The maximum value is 64.  The value
# specified must be greater than the value specified for the
# ibm-idsFSInterval attribute.  However, a value of 0 can be specified
# to indicate that no slow scan processing should be performed.  Only
# one policy rule in the set of rules for a given stack can specify
# this global scan value.
# The format of this attribute is as follows:
#    ibm-idsSSThreshold:<an integer number>
#             The default is 10.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsSSThreshold:25
attribute       ibm-idsSSThreshold                    cis   idsSSThreshold             32 normal

# ibm-idsSensitivity attribute specifies the sensitivity of events
# monitored for fast and slow scan attack detection.  Events that are
# monitored can be classified as normal, possibly suspicious, or very
# suspicious.  This attribute selects which of these types of events
# should be counted for scan attack detection.  Valid values are
# NONE, meaning no events are counted, HIGH, meaning all event types
# are counted, MEDIUM, meaning possibly suspicious and very suspicious
# events are counted, or LOW, meaning only very suspicious events are
# counted.
# The format of this attribute is as follows:
#    ibm-idsSensitivity:NONE | HIGH | MEDIUM | LOW
#             The default is NONE.
# This attribute applies to version 3 policies.
#   This is a single-valued attribute.
#    an example:
#       ibm-idsSensitivity:MEDIUM
attribute       ibm-idsSensitivity                    cis   idsSensitivity             32 normal
```

```
# ibm-idsScanExclusion attribute specifies IP addresses and optionally
# ports that are to be excluded when monitoring for scan attacks.  For
# example, responses from name servers may appear to be scan attacks,
# unless the name servers are excluded using this attribute.
# The format of this attribute is as follows:
#    ibm-idsScanExclusion:1-<RemoteIPv4Address>-<PrefixMaskLength>
#                           [-<RemoteFromPort>][-<RemoteToPort>]
#                              PrefixMaskLength is the number of unmasked
#                              leading bits.  An IP packet matches the action
#                              if its remote address unmasked bits are
#                              identical to the unmasked bits defined.
# This attribute applies to version 3 policies.
#   This is a multi-valued attribute.
#    some examples:
#       ibm-idsScanExclusion:1-130.0.1.1
#       ibm-idsScanExclusion:1-130.0.2.1-100-110
#       ibm-idsScanExclusion:1-130.0.3.1-53
#          The first example shows only an IP address, the second shows
#          a port range, and the last shows only one port, no range.
attribute       ibm-idsScanExclusion                  cis   idsScanExclusion            32 normal

# ibm-policyRepositoryName attribute specifies the user friendly name of
# an ibm-policyRepository object.  This attribute applies to version 3
# policies.
#   This is a single-valued attribute.
attribute       ibm-policyRepositoryName              cis   policyRepName              256 normal

# ibm-policySubtreesAuxContainedSet attribute provides an unordered
# set of distinguished name pointers to one or more directory subtrees
# that anchor policy-related objects.  This allows a more efficient
# retrieval of policy objects from an LDAP server.  This attribute
# applies to version 3 policies.
#   This is a multi-valued attribute.  Its value is the distinguished
#   name of the referenced directory subtree.
attribute       ibm-policySubtreesAuxContainedSet      dn    policySubtreeSet           256 normal

# SubnetAddr attribute specifies the interface for which the Type of
# Service (TOS) byte mappings defined with the SetSubnetPrioTosMask
# object are to be applied.  The specified value must be a valid interface
# for the stack for which this attribute applies.  Either an IPv4 address
# or an interface name can be specified - the only way to specify IPv6
# interfaces is by name.  The format of this attribute is as follows:
#    SubnetAddr:<IPv4Address | interface_name>
#             Default is all interfaces.
#   This is a single-valued attribute.
attribute       SubnetAddr                            cis   SubnetAddr                  32 normal

# SubnetTosMask attribute specifies the type of service (TOS) byte bits
# that are to be considered for mapping to outbound interface priorities
# using the SetSubnetPrioTosMask object.  This attribute is required.
# The format of this attribute is as follows:
#    SubnetTosMask:<a string of 8 '0' and '1'>
#   This is a single-valued attribute.
attribute       SubnetTosMask                         cis   SubnetTosMask                8 normal

# PriorityTosMapping attribute specifies type of service (TOS) byte to
# outbound interface priority mappings for the SetSubnetPrioTosMask
# object.  For devices that support tagging of Virtual LAN (VLAN) frames,
# the VLAN user priority can optionally be specified.  This attribute
# can be repeated for each such mapping to be defined.
# The format of this attribute is as follows:
#    PriorityTosMapping:<integer number-string of 8 '0' and '1'[-integer number]>
#       The following example shows the default values for the mapping:
#          PriorityTosMapping:1-11100000
#          PriorityTosMapping:1-11000000
#          PriorityTosMapping:1-10100000
#          PriorityTosMapping:1-10000000
#          PriorityTosMapping:2-01100000
#          PriorityTosMapping:3-01000000
#          PriorityTosMapping:4-00100000
#          PriorityTosMapping:4-00000000
#       The following example shows VLAN user priority specified:
#          PriorityTosMapping:1-11000000-3
#   This is a multi-valued attribute.
attribute       PriorityTosMapping                    cis   PriorityTosMap             32 normal
```

Figure 64. PAGENTAT sample

PAGENTOC sample

```
#
#  pagent_oc.conf
#
#  This file contains objectclass definitions to be defined in
#  an LDAP server for Quality of Service (QOS) and Intrusion Detection
#  Services (IDS) policies.
#

# The ibm-policy object class is an abstract class which is used as the
# root of all policy related structural classes.  This class applies to
# version 3 policies.
objectclass   ibm-policy
        requires
                objectClass
        allows
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyGroup object class is a structural subclass of
# ibm-policy that acts as a container for either a set of related
# policy rules or a set of related policy groups (e.g., groups imbedded
# within a group).  An ibm-policyGroup object can use either the
# ibm-policyRulesAuxContainedSet or ibm-policyGroupsAuxContainedSet
# pointer to realize this containment.
objectclass   ibm-policyGroup
        requires
                objectClass,
                ibm-policyGroupName
        allows
                ibm-policyGroupsAuxContainedSet,
                ibm-policyRulesAuxContainedSet,
                ibm-policyGroupKeywords,
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyRule object class is a structural subclass of
# ibm-policy that represents the "If Condition then Action" semantic
# associated with a policy.  The set of conditions (e.g., source IP
# address ranges, source port numbers etc.) are either included directly
# into an ibm-policyRule object (i.e., a simple rule) or pointed to by
# the ibm-policyRuleConditionList or ibm-policyRuleConditionListDN
# attribute (i.e., a complex rule).
objectclass   ibm-policyRule
        requires
                objectClass,
                ibm-policyRuleName
        allows
                ibm-policyRuleEnabled,
                ibm-policyRuleConditionListType,
                ibm-policyRuleConditionList,
                ibm-policyRuleConditionListDN,
                ibm-policyRuleActionList,
                ibm-policyRuleActionListDN,
                ibm-policyRuleValidityPeriodList,
                ibm-policyRuleKeywords,
                ibm-policyRuleUsage,
                ibm-policyRulePriority,
                ibm-policyRuleMandatory,
                ibm-policyRuleSequencedActions,
                ibm-policyRoles,
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyRuleConditionAssociation object class is a structural
# subclass of ibm-policy that represents policy condition objects.  The
# policy conditions themselves are represented by auxiliary subclasses
# of the auxiliary class ibm-policyConditionAuxClass.  These auxiliary
# classes are attached directly to instances of the class
# ibm-policyRuleConditionAssociation for rule-specific conditions.  For
# reusable conditions, the auxiliary classes are attached to instances
# of the class ibm-policyInstance or ibm-policyConditionInstance.  This
# class applies to version 3 policies.
objectclass   ibm-policyRuleConditionAssociation
        requires
                objectClass,
                ibm-policyConditionName,
                ibm-policyConditionGroupNumber,
                ibm-policyConditionNegated
        allows
                ibm-policyConditionDN,
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyRuleActionAssociation object class is a structural
# subclass of ibm-policy that represents policy action objects.  The
# policy actions themselves are represented by auxiliary subclasses of
# the auxiliary class ibm-policyActionAuxClass.  These auxiliary classes
# are attached directly to instances of the class
# ibm-policyRuleActionAssociation for rule-specific actions.  For
# reusable actions, the auxiliary classes are attached to instances of
# the class ibm-policyInstance or ibm-policyActionInstance.  This class
# applies to version 3 policies.
objectclass   ibm-policyRuleActionAssociation
        requires
                objectClass,
                ibm-policyActionName,
                ibm-policyActionOrder
        allows
                ibm-policyActionDN,
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyInstance object class is a structural subclass of
# ibm-policy that represents either policy condition or policy action
# objects.  The policy conditions or actions themselves are represented
# by auxiliary subclasses of the auxiliary class
# ibm-policyConditionAuxClass or ibm-policyActionAuxClass.  These
# auxiliary classes are attached directly to instances of the class
# ibm-policyRuleConditionAssociation or ibm-policyRuleActionAssociation
# for rule-specific conditions or actions.  For reusable conditions or
# actions, the auxiliary classes are attached to instances of the class
# ibm-policyInstance, ibm-policyConditionInstance or
# ibm-policyActionInstance.  This class applies to version 3 policies.
objectclass   ibm-policyInstance
        requires
                objectClass
        allows
                ibm-policyInstanceName,
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyConditionInstance object class is a structural subclass
# of ibm-policyInstance that represents policy condition objects.  The
# policy conditions themselves are represented by auxiliary subclasses
# of the auxiliary class ibm-policyConditionAuxClass.  These auxiliary
# classes are attached directly to instances of the class
# ibm-policyRuleConditionAssociation for rule-specific conditions.  For
# reusable conditions, the auxiliary classes are attached to instances
# of the class ibm-policyInstance or ibm-policyConditionInstance.  This
# class applies to version 3 policies.
objectclass   ibm-policyConditionInstance
        requires
                objectClass
        allows
                ibm-policyInstanceName,
                ibm-policyConditionName,
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyActionInstance object class is a structural subclass
# of ibm-policyInstance that represents policy action objects.  The
# policy actions themselves are represented by auxiliary subclasses
# of the auxiliary class ibm-policyActionAuxClass.  These auxiliary
# classes are attached directly to instances of the class
# ibm-policyRuleActionAssociation for rule-specific actions.  For
# reusable actions, the auxiliary classes are attached to instances of
# the class ibm-policyInstance or ibm-policyActionInstance.  This class
# applies to version 3 policies.
objectclass   ibm-policyActionInstance
        requires
                objectClass
        allows
                ibm-policyInstanceName,
                ibm-policyActionName,
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyCondition object class is a structural subclass of
# ibm-policy that represents a condition to be evaluated in conjunction
# with a policy rule object (i.e., "If Condition then Action" semantic).
# The actual conditions are contained in subclasses of this class.
# This class applies to version 2 policies.
objectclass   ibm-policyCondition
        requires
                objectClass,
                ibm-policyConditionName
        allows
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyTimePeriodCondition object class is a structural
# subclass of ibm-policyCondition that represents the time periods
# during which a policy rule is active, to be evaluated in conjunction
# with a policy rule.  The ibm-policyTimePeriodCondition object can only
# be referenced within a policy rule object.  This class applies to
# version 2 policies.
objectclass   ibm-policyTimePeriodCondition
        requires
                objectClass,
                ibm-policyConditionName
        allows
                ibm-ptpConditionTime,
                ibm-ptpConditionMonthOfYearMask,
                ibm-ptpConditionDayOfMonthMask,
                ibm-ptpConditionDayOfWeekMask,
                ibm-ptpConditionTimeOfDayMask,
                ibm-ptpConditionTimeZone,
                cn,
                ibm-policyKeywords,
                description

# The ibm-networkingPolicyCondition object class is a structural subclass
# of ibm-policyCondition that represents a set of networking related
# conditions to be evaluated in conjunction with a policy rule object.
# The networking conditions themselves are represented by the auxiliary
# subclasses ibm-routeConditionAuxClass, ibm-hostConditionAuxClass, and
# ibm-applicationConditionAuxClass, which are attached to this class.
# This class applies to version 2 policies.
objectclass   ibm-networkingPolicyCondition
        requires
                objectClass,
                ibm-policyConditionName
        allows
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyAction object class is a structural subclass of
# ibm-policy that represents an action to be performed as a result of
# evaluation of a policy rule (e.g., the "If Condition then Action"
# representation).  The actions themselves are contained in the
# ibm-serviceCategories subclass.  This class applies to version 2
# policies.
objectclass   ibm-policyAction
        requires
                objectClass,
                ibm-policyActionName
        allows
                cn,
                ibm-policyKeywords,
                description

# The ibm-serviceCategories object class is a structural subclass of
# ibm-policyAction that contains a set of Quality of Service (QoS)
# attributes to apply to a flow of IP packets, identified by a policy
# rule condition, as they traverse the network.  This class applies to
# version 2 policies.
objectclass   ibm-serviceCategories
        requires
                objectClass,
                ibm-policyActionName
        allows
                ibm-PolicyScope,
                ibm-Permission,
                ibm-MaxRate,
                ibm-MinRate,
                ibm-MaxDelay,
                ibm-OutgoingTOS,
                ibm-MaxConnections,
                ibm-Interface,
                ibm-DiffServInProfileRate,
                ibm-DiffServInProfilePeakRate,
                ibm-DiffServInProfileTokenBucket,
                ibm-DiffServInProfileMaxPacketSize,
                ibm-DiffServOutProfileTransmittedTOSByte,
                ibm-DiffServExcessTrafficTreatment,
                ibm-FlowServiceType,
                ibm-MaxRatePerFlow,
                ibm-MaxTokenBucketPerFlow,
                ibm-MaxFlows,
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyElementAuxClass object class is an auxiliary subclass of
# ibm-policy that is used to "tag" an instance of a class defined outside
# the realm of policy as being nevertheless relevant to a policy
# specification.  Every instance to which this class is attached becomes
# an instance of the ibm-policy class.  This class applies to version 3
# policies.
objectclass   ibm-policyElementAuxClass
        requires
                objectClass
        allows
                cn,
                ibm-policyKeywords,
                description

# The ibm-policyConditionAuxClass object class is an auxiliary class that
# represents a condition to be evaluated in conjunction with a policy
# rule object (i.e., "If Condition then Action" semantic).  It is
# attached directly to an instance of ibm-policyRuleConditionAssociation
# or ibm-policyRule for rule-specific conditions, or to an instance of
# ibm-policyInstance or ibm-policyConditionInstance for reusable
# conditions.  The actual conditions are represented by auxiliary
# subclasses of this class.  This class applies to version 3 policies.
objectclass   ibm-policyConditionAuxClass
        requires
                objectClass

# The ibm-policyTimePeriodConditionAuxClass object class is an
# auxiliary subclass of ibm-policyConditionAuxClass that represents the
# time periods during which a policy rule is active, to be evaluated in
# conjunction with a policy rule.  This class applies to version 3
# policies.
objectclass   ibm-policyTimePeriodConditionAuxClass
        requires
                objectClass
        allows
                ibm-ptpConditionTime,
                ibm-ptpConditionMonthOfYearMask,
                ibm-ptpConditionDayOfMonthMask,
                ibm-ptpConditionDayOfWeekMask,
                ibm-ptpConditionTimeOfDayMask,
                ibm-ptpConditionTimeZone,
                ibm-ptpConditionLocalOrUtcTime

# The ibm-networkingPolicyConditionAuxClass object class is an auxiliary
# subclass of ibm-policyConditionAuxClass that represents a set of
# networking related conditions to be evaluated in conjunction with a
# policy rule object.  The networking conditions themselves are
# represented by the auxiliary subclasses ibm-routeConditionAuxClass,
# ibm-hostConditionAuxClass, and ibm-applicationConditionAuxClass.
# This class applies to version 3 policies.
objectclass   ibm-networkingPolicyConditionAuxClass
        requires
                objectClass

# The ibm-routeConditionAuxClass object class is an auxiliary subclass
# of ibm-networkingPolicyConditionAuxClass that represents the routing
# of an entity (e.g., a packet) to be evaluated in conjunction with a
# policy rule.
objectclass   ibm-routeConditionAuxClass
```

```
        requires
                objectClass
        allows
                ibm-interface

# The ibm-ToSConditionAuxClass object class is an auxiliary subclass
# of ibm-routeConditionAuxClass that contains Type of Service (ToS) or
# Differentiated Services (DS) field parameters to be evaluated as part
# of a policy rule.
objectclass   ibm-ToSConditionAuxClass
        requires
                objectClass
        allows
                ibm-IncomingTOS

# The ibm-hostConditionAuxClass object class is an auxiliary subclass
# of ibm-networkingPolicyConditionAuxClass that represents the
# communicating end hosts (e.g., IP addresses) to be evaluated in
# conjunction with a policy rule.
objectclass   ibm-hostConditionAuxClass
        requires
                objectClass
        allows
                ibm-sourceIPAddressRange,
                ibm-destinationIPAddressRange,
                ibm-serverDomainName

# The ibm-applicationConditionAuxClass object class is an auxiliary
# subclass of ibm-networkingPolicyConditionAuxClass that represents the
# nature of the application (e.g., port 21, FTPD) and the transport
# entity (e.g., TCP) to be evaluated in conjunction with a policy rule.
objectclass   ibm-applicationConditionAuxClass
        requires
                objectClass
        allows
                ibm-sourcePortRange,
                ibm-destinationPortRange,
                ibm-protocolNumberRange,
                ibm-applicationName,
                ibm-applicationData,
                ibm-applicationPriority

# The ibm-userconditionAuxClass object class is an auxiliary
# subclass of ibm-networkingPolicyConditionAuxClass that represents the
# characteristics of the user that requests the service.
objectclass   ibm-userConditionAuxClass
        requires
                objectClass
        allows
                ibm-userNameId,
                ibm-userQoSGroup

# The ibm-idsConditionAuxClass object class is an auxiliary subclass of
# ibm-policyConditionAuxClass.  It represents conditions that must be
# true for Intrusion Detection Services (IDS) policy rules.  This class
# applies to version 3 policies.
objectclass   ibm-idsConditionAuxClass
        requires
                objectClass,
                ibm-idsConditionType
        allows
                description

# The ibm-idsAttackConditionAuxClass object class is an auxiliary
# subclass of ibm-idsConditionAuxClass representing the known types of
# intrusions to be evaluated in conjunction with an IDS policy rule.
# This class applies to version 3 policies.
```

```
objectclass   ibm-idsAttackConditionAuxClass
        requires
                objectClass
        allows
                ibm-idsAttackType,
                description

# The ibm-idsIPAttackConditionAuxClass object class is an auxiliary
# subclass of ibm-idsAttackConditionAuxClass representing allowed IP
# values for IDS IP attacks.  This class applies to version 3 policies.
objectclass   ibm-idsIPAttackConditionAuxClass
        requires
                objectClass
        allows
                ibm-idsIPOptionRange,
                description

# The ibm-idsTrafficRegulationConditionAuxClass object class is an
# auxiliary subclass of ibm-idsConditionAuxClass representing traffic
# regulation conditions.  This auxiliary class has no significant
# attributes but its inclusion in the policy condition object signifies
# that traffic regulation parameters may be provided in the
# ibm-idsTrafficRegulationActionAuxClass.  This class applies to version
# 3 policies.
objectclass   ibm-idsTrafficRegulationConditionAuxClass
        requires
                objectClass
        allows
                description

# The ibm-idsScanConditionAuxClass object class is an auxiliary subclass
# of ibm-idsConditionAuxClass representing global conditions for setting
# scanning attack detection parameters.  This auxiliary class has no
# significant attributes but its inclusion in the policy condition
# object signifies that the global scan parameters may be provided in
# the ibm-idsScanActionAuxClass.  This class applies to version 3
# policies.
objectclass   ibm-idsScanConditionAuxClass
        requires
                objectClass
        allows
                description


# The ibm-idsScanEventConditionAuxClass object class is an auxiliary
# subclass of ibm-idsConditionAuxClass specifying the scan event
# conditions to be detected.  This auxiliary class has no significant
# attributes but its inclusion in the policy condition object signifies
# that the scan event parameters may be provided in the
# ibm-idsScanSensitivityActionAuxClass and/or
# ibm-idsScanExclusionActionAuxClass.  Multiple scan events to be
# detected can be specified for a TCP/IP stack.  This class applies to
# version 3 policies.
objectclass   ibm-idsScanEventConditionAuxClass
        requires
                objectClass
        allows
                description

# The ibm-idsTransportConditionAuxClass object class is an auxiliary
# subclass of ibm-idsConditionAuxClass representing local and remote port
# ranges and protocol ranges to be applied to IDS conditions.  This class
# applies to version 3 policies.
objectclass   ibm-idsTransportConditionAuxClass
        requires
                objectClass
        allows
```

```
                ibm-idsLocalPortRange,
                ibm-idsRemotePortRange,
                ibm-idsProtocolRange,
                description

# The ibm-idsHostConditionAuxClass object class is an auxiliary subclass
# of ibm-idsConditionAuxClass representing local and remote IP hosts
# to be applied to IDS conditions.  This class applies to version 3
# policies.
objectclass   ibm-idsHostConditionAuxClass
        requires
                objectClass
        allows
                ibm-idsLocalHostIPAddress,
                ibm-idsRemoteHostIPAddress,
                description

# The ibm-policyActionAuxClass object class is an auxiliary class that
# represents an action to be performed as a result of evaluation of a
# policy rule (e.g., the "If Condition then Action" semantic).  It is
# attached directly to an instance of ibm-policyRuleActionAssociation
# for rule-specific actions, or to an instance of ibm-policyInstance or
# ibm-policyActionInstance for reusable actions.  The actions
# themselves are represented by auxiliary subclasses such as
# ibm-serviceCategoriesAuxClass.  This class applies to version 3
# policies.
objectclass   ibm-policyActionAuxClass
        requires
                objectClass

# The ibm-serviceCategoriesAuxClass object class is an auxiliary subclass
# of ibm-policyActionAuxClass that contains a set of Quality of Service
# (QoS) attributes to apply to a flow of IP packets, identified by a
# policy rule condition, as they traverse the network.  This class
# applies to version 3 policies.
objectclass   ibm-serviceCategoriesAuxClass
        requires
                objectClass
        allows
                ibm-PolicyScope,
                ibm-Permission,
                ibm-MaxRate,
                ibm-MinRate,
                ibm-MaxDelay,
                ibm-OutgoingTOS,
                ibm-MaxConnections,
                ibm-Interface,
                ibm-DiffServInProfileRate,
                ibm-DiffServInProfilePeakRate,
                ibm-DiffServInProfileTokenBucket,
                ibm-DiffServInProfileMaxPacketSize,
                ibm-DiffServOutProfileTransmittedTOSByte,
                ibm-DiffServExcessTrafficTreatment,
                ibm-FlowServiceType,
                ibm-MaxRatePerFlow,
                ibm-MaxTokenBucketPerFlow,
                ibm-MaxFlows,
                ibm-SignalClient

# The ibm-inboundConnectionAuxClass object class is an auxiliary subclass
# of ibm-policyActionAuxClass that contains a set of Quality of Service
# (QoS) attributes to apply to an inbound flow of IP packets, identified
# by a policy rule condition, as they traverse the network.  This class
# applies to version 3 policies.
objectclass   ibm-inboundConnectionAuxClass
        requires
                objectClass
        allows
                ibm-inboundScope,
                ibm-averageConnectionRate,
                ibm-peakConnectionRate,
                ibm-connectionBurstSize,
                ibm-averageApplicationRequestRate,
                ibm-applicationRequestPeakRate,
                ibm-applicationRequestBurstSize,
                ibm-prioritizedQueue

# The ibm-idsActionAuxClass object class is an auxiliary subclass of
# ibm-policyActionAuxClass.  It represents actions to be performed
# for a corresponding Intrusion Detection Services (IDS) rule.  This
# class applies to version 3 policies.
objectclass   ibm-idsActionAuxClass
        requires
                objectClass,
                ibm-idsActionType
        allows
                description

# The ibm-idsNotificationAuxClass object class is an auxiliary subclass
# of ibm-idsActionAuxClass indicating IDS notification actions.  This
# class applies to version 3 policies.
objectclass   ibm-idsNotificationAuxClass
        requires
                objectClass
        allows
                ibm-idsNotification,
                ibm-idsStatInterval,
                ibm-idsLoggingLevel,
                ibm-idsTypeActions,
                ibm-idsTraceData,
                ibm-idsTraceRecordSize,
                description

# The ibm-idsAttackActionsAuxClass object class is an auxiliary subclass
# of ibm-idsActionAuxClass indicating IDS attack type actions.  This
# class applies to version 3 policies.
objectclass   ibm-idsAttackActionsAuxClass
        requires
                objectClass
        allows
                ibm-idsMaxEventMessage,
                description

# The ibm-idsFloodAttackActionsAuxClass object class is an auxiliary
# subclass of ibm-idsAttackActionsAuxClass indicating IDS flood attack
# type actions. This class applies to version 3 policies.
objectclass   ibm-idsFloodAttackActionsAuxClass
        requires
                objectClass
        allows
                ibm-idsIfcFloodPercentage,
                ibm-idsIfcFloodMinDiscard,
                description

# The ibm-idsTrafficRegulationActionAuxClass object class is an
# auxiliary subclass of ibm-idsActionAuxClass representing actions for
# handling Traffic Regulation.  It has no significant attributes but
# is used to anchor additional traffic regulation subclasses.  This
# class applies to version 3 policies.
objectclass   ibm-idsTrafficRegulationActionAuxClass
        requires
                objectClass
        allows
                description
```

```
# The ibm-idsTRtcpActionAuxClass object class is an auxiliary subclass
# of ibm-idsTrafficRegulationActionAuxClass representing actions for
# handling traffic regulation for TCP on a per port basis.  This class
# applies to version 3 policies.
objectclass   ibm-idsTRtcpActionAuxClass
        requires
                objectClass
        allows
                ibm-idsTRtcpTotalConnections,
                ibm-idsTRtcpPercentage,
                ibm-idsTRtcpLimitScope,
                description

# The idsTRudpActionAuxClass object class is an auxiliary subclass of
# ibm-idsTrafficRegulationActionAuxClass representing actions for
# handling traffic regulation for UDP.  This class applies to version 3
# policies.
objectclass   ibm-idsTRudpActionAuxClass
        requires
                objectClass
        allows
                ibm-idsTRudpQueueSize,
                description

# The ibm-idsScanActionAuxClass object class is an auxiliary subclass
# of ibm-idsActionAuxClass representing the global setting of scan
# attack detection parameters.  Note that only one set of these
# parameters is allowed per TCP/IP stack.  This class applies to version
# 3 policies.
objectclass   ibm-idsScanActionAuxClass
        requires
                objectClass
        allows
                ibm-idsFSInterval,
                ibm-idsFSThreshold,
                ibm-idsSSInterval,
                ibm-idsSSThreshold,
                description

# The ibm-idsScanSensitivityActionAuxClass object class is an
# auxiliary subclass of ibm-idsActionAuxClass representing the
# sensitivity of the scan events which are detected.  This class
# applies to version 3 policies.
objectclass   ibm-idsScanSensitivityActionAuxClass
        requires
                objectClass
        allows
                ibm-idsSensitivity,
                description

# The ibm-idsScanExclusionActionAuxClass object class is an
# auxiliary subclass of ibm-idsActionAuxClass representing exclusions
# in conjunction with scanning attacks.  It is valid to be attached to
# an IDS action when the corresponding condition is for detecting scan
# events.  This class applies to version 3 policies.
objectclass   ibm-idsScanExclusionActionAuxClass
        requires
                objectClass
        allows
                ibm-idsScanExclusion,
                description

# The ibm-policyRepository object class is a structural class which is
# used as the root of reusable policy information, such as policy
# conditions and policy actions.  This class applies to version 3
# policies.
```

```
objectclass   ibm-policyRepository
        requires
                objectClass,
                ibm-policyRepositoryName
        allows
                cn,
                description

# The ibm-policySubtreesPtrAuxClass object class is an auxiliary class
# that allows a set of pointers to be defined which point to sets of
# objects that are at the root of subtrees containing policy-related
# information.  By attaching this pointer attribute to instances of
# various other classes, a policy administrator has a flexible way of
# providing an entry point into the directory that allows a client to
# locate and retrieve the policy information relevant to it in an
# efficient manner.  This class applies to version 3 policies.
objectclass   ibm-policySubtreesPtrAuxClass
        requires
                objectClass
        allows
                ibm-policySubtreesAuxContainedSet

# The ibm-policyGroupContainmentAuxClass object class is an auxiliary
# class used to bind policy group objects to an appropriate container
# object (e.g., another policy group object) via its attribute pointer
# ibm-policyGroupsAuxContainedSet.  It is attached to instances of
# ibm-policyGroup.
objectclass   ibm-policyGroupContainmentAuxClass
        requires
                objectClass
        allows
                ibm-policyGroupsAuxContainedSet

# The ibm-policyRuleContainmentAuxClass object class is an auxiliary
# class used to bind policy rule objects to an appropriate container
# object (e.g., a # policy group object) via its attribute pointer
# ibm-policyRulesAuxContainedSet.  It is attached to instances of
# ibm-policyGroup.
objectclass   ibm-policyRuleContainmentAuxClass
        requires
                objectClass
        allows
                ibm-policyRulesAuxContainedSet

# The ibm-policyGroupLoadDistributionAuxClass object class is an
# auxiliary class used to specify load distribution attributes for
# policy rules contained in the policy group.  It is attached to
# instances of ibm-policyGroup.  This class applies to version 2
# policies.
objectclass   ibm-policyGroupLoadDistributionAuxClass
        requires
                objectClass
        allows
                ibm-policyGroupForLoadDistribution

# The SetSubnetPrioTosMask object class is a structural class that
# defines a mapping of IP type of service (TOS) byte to outbound
# interface priority values.  It is not directly related to the other
# object classes defined for policy groups, rules, conditions, or
# actions.
objectclass   SetSubnetPrioTosMask
        requires
                objectClass,
                SubnetTosMask
        allows
```

```
                                    cn,
                                    SubnetAddr,
                                    PriorityTosMapping,
                                    description
```

Appendix C. Related protocol specifications

This appendix lists the related protocol specifications (RFCs) for TCP/IP. The Internet Protocol suite is still evolving through requests for comments (RFC). New protocols are being designed and implemented by researchers and are brought to the attention of the Internet community in the form of RFCs. Some of these protocols are so useful that they become recommended protocols. That is, all future implementations for TCP/IP are recommended to implement these particular functions or protocols. These become the *de facto* standards, on which the TCP/IP protocol suite is built.

You can request RFCs through electronic mail, from the automated Network Information Center (NIC) mail server, by sending a message to `service@nic.ddn.mil` with a subject line of `RFC` *`nnnn`* for text versions or a subject line of `RFC` *`nnnn`*`.PS` for PostScript versions. To request a copy of the RFC index, send a message with a subject line of `RFC INDEX`.

For more information, contact `nic@nic.ddn.mil` or at:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Hard copies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available at the following Web address: http://www.rfc-editor.org/rfc.html.

Draft RFCs that have been implemented in this and previous Communications Server releases are listed at the end of this topic.

Many features of TCP/IP Services are based on the following RFCs:

RFC	Title and Author
RFC 652	*Telnet output carriage-return disposition option* D. Crocker
RFC 653	*Telnet output horizontal tabstops option* D. Crocker
RFC 654	*Telnet output horizontal tab disposition option* D. Crocker
RFC 655	*Telnet output formfeed disposition option* D. Crocker
RFC 657	*Telnet output vertical tab disposition option* D. Crocker
RFC 658	*Telnet output linefeed disposition* D. Crocker
RFC 698	*Telnet extended ASCII option* T. Mock

RFC 726
: *Remote Controlled Transmission and Echoing Telnet option* J. Postel, D. Crocker

RFC 727
: *Telnet logout option* M.R. Crispin

RFC 732
: *Telnet Data Entry Terminal option* J.D. Day

RFC 733
: *Standard for the format of ARPA network text messages* D. Crocker, J. Vittal, K.T. Pogran, D.A. Henderson

RFC 734
: *SUPDUP Protocol* M.R. Crispin

RFC 735
: *Revised Telnet byte macro option* D. Crocker, R.H. Gumpertz

RFC 736
: *Telnet SUPDUP option* M.R. Crispin

RFC 749
: *Telnet SUPDUP—Output option* B. Greenberg

RFC 765
: *File Transfer Protocol specification* J. Postel

RFC 768
: *User Datagram Protocol* J. Postel

RFC 779
: *Telnet send-location option* E. Killian

RFC 783
: *TFTP Protocol (revision 2)* K.R. Sollins

RFC 791
: *Internet Protocol* J. Postel

RFC 792
: *Internet Control Message Protocol* J. Postel

RFC 793
: *Transmission Control Protocol* J. Postel

RFC 820
: *Assigned numbers* J. Postel

RFC 821
: *Simple Mail Transfer Protocol* J. Postel

RFC 822
: *Standard for the format of ARPA Internet text messages* D. Crocker

RFC 823
: *DARPA Internet gateway* R. Hinden, A. Sheltzer

RFC 826
: *Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware* D. Plummer

RFC 854
: *Telnet Protocol Specification* J. Postel, J. Reynolds

RFC 855
Telnet Option Specification J. Postel, J. Reynolds

RFC 856
Telnet Binary Transmission J. Postel, J. Reynolds

RFC 857
Telnet Echo Option J. Postel, J. Reynolds

RFC 858
Telnet Suppress Go Ahead Option J. Postel, J. Reynolds

RFC 859
Telnet Status Option J. Postel, J. Reynolds

RFC 860
Telnet Timing Mark Option J. Postel, J. Reynolds

RFC 861
Telnet Extended Options: List Option J. Postel, J. Reynolds

RFC 862
Echo Protocol J. Postel

RFC 863
Discard Protocol J. Postel

RFC 864
Character Generator Protocol J. Postel

RFC 865
Quote of the Day Protocol J. Postel

RFC 868
Time Protocol J. Postel, K. Harrenstien

RFC 877
Standard for the transmission of IP datagrams over public data networks J.T. Korb

RFC 883
Domain names: Implementation specification P.V. Mockapetris

RFC 884
Telnet terminal type option M. Solomon, E. Wimmers

RFC 885
Telnet end of record option J. Postel

RFC 894
Standard for the transmission of IP datagrams over Ethernet networks C. Hornig

RFC 896
Congestion control in IP/TCP internetworks J. Nagle

RFC 903
Reverse Address Resolution Protocol R. Finlayson, T. Mann, J. Mogul, M. Theimer

RFC 904
Exterior Gateway Protocol formal specification D. Mills

RFC 919
Broadcasting Internet Datagrams J. Mogul

RFC 922
: *Broadcasting Internet datagrams in the presence of subnets* J. Mogul

RFC 927
: *TACACS user identification Telnet option* B.A. Anderson

RFC 933
: *Output marking Telnet option* S. Silverman

RFC 946
: *Telnet terminal location number option* R. Nedved

RFC 950
: *Internet Standard Subnetting Procedure* J. Mogul, J. Postel

RFC 952
: *DoD Internet host table specification* K. Harrenstien, M. Stahl, E. Feinler

RFC 959
: *File Transfer Protocol* J. Postel, J.K. Reynolds

RFC 961
: *Official ARPA-Internet protocols* J.K. Reynolds, J. Postel

RFC 974
: *Mail routing and the domain system* C. Partridge

RFC 1001
: *Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods* NetBios Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, End-to-End Services Task Force

RFC 1002
: *Protocol Standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications* NetBios Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, End-to-End Services Task Force

RFC 1006
: *ISO transport services on top of the TCP: Version 3* M.T. Rose, D.E. Cass

RFC 1009
: *Requirements for Internet gateways* R. Braden, J. Postel

RFC 1011
: *Official Internet protocols* J. Reynolds, J. Postel

RFC 1013
: *X Window System Protocol, version 11: Alpha update April 1987* R. Scheifler

RFC 1014
: *XDR: External Data Representation standard* Sun Microsystems

RFC 1027
: *Using ARP to implement transparent subnet gateways* S. Carl-Mitchell, J. Quarterman

RFC 1032
: *Domain administrators guide* M. Stahl

RFC 1033
: *Domain administrators operations guide* M. Lottor

RFC 1034
: *Domain names—concepts and facilities* P.V. Mockapetris

RFC 1035
Domain names—implementation and specification P.V. Mockapetris

RFC 1038
Draft revised IP security option M. St. Johns

RFC 1041
Telnet 3270 regime option Y. Rekhter

RFC 1042
Standard for the transmission of IP datagrams over IEEE 802 networks J. Postel, J. Reynolds

RFC 1043
Telnet Data Entry Terminal option: DODIIS implementation A. Yasuda, T. Thompson

RFC 1044
Internet Protocol on Network System's HYPERchannel: Protocol specification K. Hardwick, J. Lekashman

RFC 1053
Telnet X.3 PAD option S. Levy, T. Jacobson

RFC 1055
Nonstandard for transmission of IP datagrams over serial lines: SLIP J. Romkey

RFC 1057
RPC: Remote Procedure Call Protocol Specification: Version 2 Sun Microsystems

RFC 1058
Routing Information Protocol C. Hedrick

RFC 1060
Assigned numbers J. Reynolds, J. Postel

RFC 1067
Simple Network Management Protocol J.D. Case, M. Fedor, M.L. Schoffstall, J. Davin

RFC 1071
Computing the Internet checksum R.T. Braden, D.A. Borman, C. Partridge

RFC 1072
TCP extensions for long-delay paths V. Jacobson, R.T. Braden

RFC 1073
Telnet window size option D. Waitzman

RFC 1079
Telnet terminal speed option C. Hedrick

RFC 1085
ISO presentation services on top of TCP/IP based internets M.T. Rose

RFC 1091
Telnet terminal-type option J. VanBokkelen

RFC 1094
NFS: Network File System Protocol specification Sun Microsystems

RFC 1096
Telnet X display location option G. Marcy

RFC 1101
DNS encoding of network names and other types P. Mockapetris

RFC 1112
: *Host extensions for IP multicasting* S.E. Deering

RFC 1113
: *Privacy enhancement for Internet electronic mail: Part I — message encipherment and authentication procedures* J. Linn

RFC 1118
: *Hitchhikers Guide to the Internet* E. Krol

RFC 1122
: *Requirements for Internet Hosts—Communication Layers* R. Braden, Ed.

RFC 1123
: *Requirements for Internet Hosts—Application and Support* R. Braden, Ed.

RFC 1146
: *TCP alternate checksum options* J. Zweig, C. Partridge

RFC 1155
: *Structure and identification of management information for TCP/IP-based internets* M. Rose, K. McCloghrie

RFC 1156
: *Management Information Base for network management of TCP/IP-based internets* K. McCloghrie, M. Rose

RFC 1157
: *Simple Network Management Protocol (SNMP)* J. Case, M. Fedor, M. Schoffstall, J. Davin

RFC 1158
: *Management Information Base for network management of TCP/IP-based internets: MIB-II* M. Rose

RFC 1166
: *Internet numbers* S. Kirkpatrick, M.K. Stahl, M. Recker

RFC 1179
: *Line printer daemon protocol* L. McLaughlin

RFC 1180
: *TCP/IP tutorial* T. Socolofsky, C. Kale

RFC 1183
: *New DNS RR Definitions* C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris

RFC 1184
: *Telnet Linemode Option* D. Borman

RFC 1186
: *MD4 Message Digest Algorithm* R.L. Rivest

RFC 1187
: *Bulk Table Retrieval with the SNMP* M. Rose, K. McCloghrie, J. Davin

RFC 1188
: *Proposed Standard for the Transmission of IP Datagrams over FDDI Networks* D. Katz

RFC 1190
: *Experimental Internet Stream Protocol: Version 2 (ST-II)* C. Topolcic

RFC 1191
Path MTU discovery J. Mogul, S. Deering

RFC 1198
FYI on the X window system R. Scheifler

RFC 1207
FYI on Questions and Answers: Answers to commonly asked "experienced Internet user" questions G. Malkin, A. Marine, J. Reynolds

RFC 1208
Glossary of networking terms O. Jacobsen, D. Lynch

RFC 1213
Management Information Base for Network Management of TCP/IP-based internets: MIB-II K. McCloghrie, M.T. Rose

RFC 1215
Convention for defining traps for use with the SNMP M. Rose

RFC 1227
SNMP MUX protocol and MIB M.T. Rose

RFC 1228
SNMP-DPI: Simple Network Management Protocol Distributed Program Interface G. Carpenter, B. Wijnen

RFC 1229
Extensions to the generic-interface MIB K. McCloghrie

RFC 1230
IEEE 802.4 Token Bus MIB K. McCloghrie, R. Fox

RFC 1231
IEEE 802.5 Token Ring MIB K. McCloghrie, R. Fox, E. Decker

RFC 1236
IP to X.121 address mapping for DDN L. Morales, P. Hasse

RFC 1256
ICMP Router Discovery Messages S. Deering, Ed.

RFC 1267
Border Gateway Protocol 3 (BGP-3) K. Lougheed, Y. Rekhter

RFC 1268
Application of the Border Gateway Protocol in the Internet Y. Rekhter, P. Gross

RFC 1269
Definitions of Managed Objects for the Border Gateway Protocol: Version 3 S. Willis, J. Burruss

RFC 1270
SNMP Communications Services F. Kastenholz, ed.

RFC 1285
FDDI Management Information Base J. Case

RFC 1315
Management Information Base for Frame Relay DTEs C. Brown, F. Baker, C. Carvalho

RFC 1321
The MD5 Message-Digest Algorithm R. Rivest

RFC 1323
: *TCP Extensions for High Performance* V. Jacobson, R. Braden, D. Borman

RFC 1325
: *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* G. Malkin, A. Marine

RFC 1327
: *Mapping between X.400 (1988)/ISO 10021 and RFC 822* S. Hardcastle-Kille

RFC 1340
: *Assigned Numbers* J. Reynolds, J. Postel

RFC 1344
: *Implications of MIME for Internet Mail Gateways* N. Bornstein

RFC 1349
: *Type of Service in the Internet Protocol Suite* P. Almquist

RFC 1350
: *The TFTP Protocol (Revision 2)* K.R. Sollins

RFC 1351
: *SNMP Administrative Model* J. Davin, J. Galvin, K. McCloghrie

RFC 1352
: *SNMP Security Protocols* J. Galvin, K. McCloghrie, J. Davin

RFC 1353
: *Definitions of Managed Objects for Administration of SNMP Parties* K. McCloghrie, J. Davin, J. Galvin

RFC 1354
: *IP Forwarding Table MIB* F. Baker

RFC 1356
: *Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode* A. Malis, D. Robinson, R. Ullmann

RFC 1358
: *Charter of the Internet Architecture Board (IAB)* L. Chapin

RFC 1363
: *A Proposed Flow Specification* C. Partridge

RFC 1368
: *Definition of Managed Objects for IEEE 802.3 Repeater Devices* D. McMaster, K. McCloghrie

RFC 1372
: *Telnet Remote Flow Control Option* C. L. Hedrick, D. Borman

RFC 1374
: *IP and ARP on HIPPI* J. Renwick, A. Nicholson

RFC 1381
: *SNMP MIB Extension for X.25 LAPB* D. Throop, F. Baker

RFC 1382
: *SNMP MIB Extension for the X.25 Packet Layer* D. Throop

RFC 1387
: *RIP Version 2 Protocol Analysis* G. Malkin

RFC 1388
: *RIP Version 2 Carrying Additional Information* G. Malkin

RFC 1389
: *RIP Version 2 MIB Extensions* G. Malkin, F. Baker

RFC 1390
: *Transmission of IP and ARP over FDDI Networks* D. Katz

RFC 1393
: *Traceroute Using an IP Option* G. Malkin

RFC 1398
: *Definitions of Managed Objects for the Ethernet-Like Interface Types* F. Kastenholz

RFC 1408
: *Telnet Environment Option* D. Borman, Ed.

RFC 1413
: *Identification Protocol* M. St. Johns

RFC 1416
: *Telnet Authentication Option* D. Borman, ed.

RFC 1420
: *SNMP over IPX* S. Bostock

RFC 1428
: *Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME* G. Vaudreuil

RFC 1442
: *Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1443
: *Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1445
: *Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Galvin, K. McCloghrie

RFC 1447
: *Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)* K. McCloghrie, J. Galvin

RFC 1448
: *Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1464
: *Using the Domain Name System to Store Arbitrary String Attributes* R. Rosenbaum

RFC 1469
: *IP Multicast over Token-Ring Local Area Networks* T. Pusateri

RFC 1483
: *Multiprotocol Encapsulation over ATM Adaptation Layer 5* Juha Heinanen

RFC 1514
: *Host Resources MIB* P. Grillo, S. Waldbusser

RFC 1516
: *Definitions of Managed Objects for IEEE 802.3 Repeater Devices* D. McMaster, K. McCloghrie

RFC 1521
MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies N. Borenstein, N. Freed

RFC 1535
A Security Problem and Proposed Correction With Widely Deployed DNS Software E. Gavron

RFC 1536
Common DNS Implementation Errors and Suggested Fixes A. Kumar, J. Postel, C. Neuman, P. Danzig, S. Miller

RFC 1537
Common DNS Data File Configuration Errors P. Beertema

RFC 1540
Internet Official Protocol Standards J. Postel

RFC 1571
Telnet Environment Option Interoperability Issues D. Borman

RFC 1572
Telnet Environment Option S. Alexander

RFC 1573
Evolution of the Interfaces Group of MIB-II K. McCloghrie, F. Kastenholz

RFC 1577
Classical IP and ARP over ATM M. Laubach

RFC 1583
OSPF Version 2 J. Moy

RFC 1591
Domain Name System Structure and Delegation J. Postel

RFC 1592
Simple Network Management Protocol Distributed Protocol Interface Version 2.0 B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters

RFC 1594
FYI on Questions and Answers— Answers to Commonly Asked "New Internet User" Questions A. Marine, J. Reynolds, G. Malkin

RFC 1644
T/TCP — TCP Extensions for Transactions Functional Specification R. Braden

RFC 1646
TN3270 Extensions for LUname and Printer Selection C. Graves, T. Butts, M. Angel

RFC 1647
TN3270 Enhancements B. Kelly

RFC 1652
SMTP Service Extension for 8bit-MIMEtransport J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker

RFC 1664
Using the Internet DNS to Distribute RFC1327 Mail Address Mapping Tables C. Allochio, A. Bonito, B. Cole, S. Giordano, R. Hagens

RFC 1693
An Extension to TCP: Partial Order Service T. Connolly, P. Amer, P. Conrad

RFC 1695
Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2 M. Ahmed, K. Tesink

RFC 1701
Generic Routing Encapsulation (GRE) S. Hanks, T. Li, D. Farinacci, P. Traina

RFC 1702
Generic Routing Encapsulation over IPv4 networks S. Hanks, T. Li, D. Farinacci, P. Traina

RFC 1706
DNS NSAP Resource Records B. Manning, R. Colella

RFC 1712
DNS Encoding of Geographical Location C. Farrell, M. Schulze, S. Pleitner D. Baldoni

RFC 1713
Tools for DNS debugging A. Romao

RFC 1723
RIP Version 2—Carrying Additional Information G. Malkin

RFC 1752
The Recommendation for the IP Next Generation Protocol S. Bradner, A. Mankin

RFC 1766
Tags for the Identification of Languages H. Alvestrand

RFC 1771
A Border Gateway Protocol 4 (BGP-4) Y. Rekhter, T. Li

RFC 1794
DNS Support for Load Balancing T. Brisco

RFC 1819
Internet Stream Protocol Version 2 (ST2) Protocol Specification—Version ST2+ L. Delgrossi, L. Berger Eds.

RFC 1826
IP Authentication Header R. Atkinson

RFC 1828
IP Authentication using Keyed MD5 P. Metzger, W. Simpson

RFC 1829
The ESP DES-CBC Transform P. Karn, P. Metzger, W. Simpson

RFC 1830
SMTP Service Extensions for Transmission of Large and Binary MIME Messages G. Vaudreuil

RFC 1831
RPC: Remote Procedure Call Protocol Specification Version 2 R. Srinivasan

RFC 1832
XDR: External Data Representation Standard R. Srinivasan

RFC 1833
Binding Protocols for ONC RPC Version 2 R. Srinivasan

RFC 1850
OSPF Version 2 Management Information Base F. Baker, R. Coltun

RFC 1854
: *SMTP Service Extension for Command Pipelining* N. Freed

RFC 1869
: *SMTP Service Extensions* J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker

RFC 1870
: *SMTP Service Extension for Message Size Declaration* J. Klensin, N. Freed, K. Moore

RFC 1876
: *A Means for Expressing Location Information in the Domain Name System* C. Davis, P. Vixie, T. Goodwin, I. Dickinson

RFC 1883
: *Internet Protocol, Version 6 (IPv6) Specification* S. Deering, R. Hinden

RFC 1884
: *IP Version 6 Addressing Architecture* R. Hinden, S. Deering, Eds.

RFC 1886
: *DNS Extensions to support IP version 6* S. Thomson, C. Huitema

RFC 1888
: *OSI NSAPs and IPv6* J. Bound, B. Carpenter, D. Harrington, J. Houldsworth, A. Lloyd

RFC 1891
: *SMTP Service Extension for Delivery Status Notifications* K. Moore

RFC 1892
: *The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages* G. Vaudreuil

RFC 1894
: *An Extensible Message Format for Delivery Status Notifications*K. Moore, G. Vaudreuil

RFC 1901
: *Introduction to Community-based SNMPv2* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1902
: *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1903
: *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1904
: *Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1905
: *Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1906
: *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1907
Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2) J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1908
Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1912
Common DNS Operational and Configuration Errors D. Barr

RFC 1918
Address Allocation for Private Internets Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot, E. Lear

RFC 1928
SOCKS Protocol Version 5 M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones

RFC 1930
Guidelines for creation, selection, and registration of an Autonomous System (AS) J. Hawkinson, T. Bates

RFC 1939
Post Office Protocol-Version 3 J. Myers, M. Rose

RFC 1981
Path MTU Discovery for IP version 6 J. McCann, S. Deering, J. Mogul

RFC 1982
Serial Number Arithmetic R. Elz, R. Bush

RFC 1985
SMTP Service Extension for Remote Message Queue Starting J. De Winter

RFC 1995
Incremental Zone Transfer in DNS M. Ohta

RFC 1996
A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY) P. Vixie

RFC 2010
Operational Criteria for Root Name Servers B. Manning, P. Vixie

RFC 2011
SNMPv2 Management Information Base for the Internet Protocol using SMIv2 K. McCloghrie, Ed.

RFC 2012
SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2 K. McCloghrie, Ed.

RFC 2013
SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2 K. McCloghrie, Ed.

RFC 2018
TCP Selective Acknowledgement Options M. Mathis, J. Mahdavi, S. Floyd, A. Romanow

RFC 2026
The Internet Standards Process — Revision 3 S. Bradner

RFC 2030
: *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI* D. Mills

RFC 2033
: *Local Mail Transfer Protocol* J. Myers

RFC 2034
: *SMTP Service Extension for Returning Enhanced Error Codes*N. Freed

RFC 2040
: *The RC5, RC5–CBC, RC-5–CBC-Pad, and RC5–CTS Algorithms*R. Baldwin, R. Rivest

RFC 2045
: *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* N. Freed, N. Borenstein

RFC 2052
: *A DNS RR for specifying the location of services (DNS SRV)* A. Gulbrandsen, P. Vixie

RFC 2065
: *Domain Name System Security Extensions* D. Eastlake 3rd, C. Kaufman

RFC 2066
: *TELNET CHARSET Option* R. Gellens

RFC 2080
: *RIPng for IPv6* G. Malkin, R. Minnear

RFC 2096
: *IP Forwarding Table MIB* F. Baker

RFC 2104
: *HMAC: Keyed-Hashing for Message Authentication* H. Krawczyk, M. Bellare, R. Canetti

RFC 2119
: *Keywords for use in RFCs to Indicate Requirement Levels* S. Bradner

RFC 2133
: *Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, W. Stevens

RFC 2136
: *Dynamic Updates in the Domain Name System (DNS UPDATE)* P. Vixie, Ed., S. Thomson, Y. Rekhter, J. Bound

RFC 2137
: *Secure Domain Name System Dynamic Update* D. Eastlake 3rd

RFC 2163
: *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)* C. Allocchio

RFC 2168
: *Resolution of Uniform Resource Identifiers using the Domain Name System* R. Daniel, M. Mealling

RFC 2178
: *OSPF Version 2* J. Moy

RFC 2181
: *Clarifications to the DNS Specification* R. Elz, R. Bush

RFC 2205
Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin

RFC 2210
The Use of RSVP with IETF Integrated Services J. Wroclawski

RFC 2211
Specification of the Controlled-Load Network Element Service J. Wroclawski

RFC 2212
Specification of Guaranteed Quality of Service S. Shenker, C. Partridge, R. Guerin

RFC 2215
General Characterization Parameters for Integrated Service Network Elements S. Shenker, J. Wroclawski

RFC 2217
Telnet Com Port Control Option G. Clarke

RFC 2219
Use of DNS Aliases for Network Services M. Hamilton, R. Wright

RFC 2228
FTP Security Extensions M. Horowitz, S. Lunt

RFC 2230
Key Exchange Delegation Record for the DNS R. Atkinson

RFC 2233
The Interfaces Group MIB using SMIv2 K. McCloghrie, F. Kastenholz

RFC 2240
A Legal Basis for Domain Name Allocation O. Vaughn

RFC 2246
The TLS Protocol Version 1.0 T. Dierks, C. Allen

RFC 2251
Lightweight Directory Access Protocol (v3) M. Wahl, T. Howes, S. Kille

RFC 2253
Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names M. Wahl, S. Kille, T. Howes

RFC 2254
The String Representation of LDAP Search Filters T. Howes

RFC 2261
An Architecture for Describing SNMP Management Frameworks D. Harrington, R. Presuhn, B. Wijnen

RFC 2262
Message Processing and Dispatching for the Simple Network Management Protocol (SNMP) J. Case, D. Harrington, R. Presuhn, B. Wijnen

RFC 2271
An Architecture for Describing SNMP Management Frameworks D. Harrington, R. Presuhn, B. Wijnen

RFC 2273
SNMPv3 Applications D. Levi, P. Meyer, B. Stewartz

RFC 2274
: *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

RFC 2275
: *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

RFC 2279
: *UTF-8, a transformation format of ISO 10646* F. Yergeau

RFC 2292
: *Advanced Sockets API for IPv6* W. Stevens, M. Thomas

RFC 2308
: *Negative Caching of DNS Queries (DNS NCACHE)* M. Andrews

RFC 2317
: *Classless IN-ADDR.ARPA delegation* H. Eidnes, G. de Groot, P. Vixie

RFC 2320
: *Definitions of Managed Objects for Classical IP and ARP Over ATM Using SMIv2 (IPOA-MIB)* M. Greene, J. Luciani, K. White, T. Kuo

RFC 2328
: *OSPF Version 2* J. Moy

RFC 2345
: *Domain Names and Company Name Retrieval* J. Klensin, T. Wolf, G. Oglesby

RFC 2352
: *A Convention for Using Legal Names as Domain Names* O. Vaughn

RFC 2355
: *TN3270 Enhancements* B. Kelly

RFC 2358
: *Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson

RFC 2373
: *IP Version 6 Addressing Architecture* R. Hinden, S. Deering

RFC 2374
: *An IPv6 Aggregatable Global Unicast Address Format* R. Hinden, M. O'Dell, S. Deering

RFC 2375
: *IPv6 Multicast Address Assignments* R. Hinden, S. Deering

RFC 2385
: *Protection of BGP Sessions via the TCP MD5 Signature Option* A. Hefferman

RFC 2389
: *Feature negotiation mechanism for the File Transfer Protocol* P. Hethmon, R. Elz

RFC 2401
: *Security Architecture for Internet Protocol* S. Kent, R. Atkinson

RFC 2402
: *IP Authentication Header* S. Kent, R. Atkinson

RFC 2403
: *The Use of HMAC-MD5–96 within ESP and AH* C. Madson, R. Glenn

RFC 2404
The Use of HMAC-SHA–1–96 within ESP and AH C. Madson, R. Glenn

RFC 2405
The ESP DES-CBC Cipher Algorithm With Explicit IV C. Madson, N. Doraswamy

RFC 2406
IP Encapsulating Security Payload (ESP) S. Kent, R. Atkinson

RFC 2407
*The Internet IP Security Domain of Interpretation for ISAKMP*D. Piper

RFC 2408
Internet Security Association and Key Management Protocol (ISAKMP) D. Maughan, M. Schertler, M. Schneider, J. Turner

RFC 2409
The Internet Key Exchange (IKE) D. Harkins, D. Carrel

RFC 2410
The NULL Encryption Algorithm and Its Use With IPsec R. Glenn, S. Kent,

RFC 2428
FTP Extensions for IPv6 and NATs M. Allman, S. Ostermann, C. Metz

RFC 2445
Internet Calendaring and Scheduling Core Object Specification (iCalendar) F. Dawson, D. Stenerson

RFC 2459
Internet X.509 Public Key Infrastructure Certificate and CRL Profile R. Housley, W. Ford, W. Polk, D. Solo

RFC 2460
Internet Protocol, Version 6 (IPv6) Specification S. Deering, R. Hinden

RFC 2461
Neighbor Discovery for IP Version 6 (IPv6) T. Narten, E. Nordmark, W. Simpson

RFC 2462
IPv6 Stateless Address Autoconfiguration S. Thomson, T. Narten

RFC 2463
Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification A. Conta, S. Deering

RFC 2464
Transmission of IPv6 Packets over Ethernet Networks M. Crawford

RFC 2466
Management Information Base for IP Version 6: ICMPv6 Group D. Haskin, S. Onishi

RFC 2476
Message Submission R. Gellens, J. Klensin

RFC 2487
SMTP Service Extension for Secure SMTP over TLS P. Hoffman

RFC 2505
Anti-Spam Recommendations for SMTP MTAs G. Lindberg

RFC 2523
Photuris: Extended Schemes and Attributes P. Karn, W. Simpson

RFC 2535
Domain Name System Security Extensions D. Eastlake 3rd

RFC 2538
Storing Certificates in the Domain Name System (DNS) D. Eastlake 3rd, O. Gudmundsson

RFC 2539
Storage of Diffie-Hellman Keys in the Domain Name System (DNS) D. Eastlake 3rd

RFC 2540
Detached Domain Name System (DNS) Information D. Eastlake 3rd

RFC 2554
SMTP Service Extension for Authentication J. Myers

RFC 2570
Introduction to Version 3 of the Internet-standard Network Management Framework J. Case, R. Mundy, D. Partain, B. Stewart

RFC 2571
An Architecture for Describing SNMP Management Frameworks B. Wijnen, D. Harrington, R. Presuhn

RFC 2572
Message Processing and Dispatching for the Simple Network Management Protocol (SNMP) J. Case, D. Harrington, R. Presuhn, B. Wijnen

RFC 2573
SNMP Applications D. Levi, P. Meyer, B. Stewart

RFC 2574
User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) U. Blumenthal, B. Wijnen

RFC 2575
View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) B. Wijnen, R. Presuhn, K. McCloghrie

RFC 2576
Co-Existence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework R. Frye, D. Levi, S. Routhier, B. Wijnen

RFC 2578
Structure of Management Information Version 2 (SMIv2) K. McCloghrie, D. Perkins, J. Schoenwaelder

RFC 2579
Textual Conventions for SMIv2 K. McCloghrie, D. Perkins, J. Schoenwaelder

RFC 2580
Conformance Statements for SMIv2 K. McCloghrie, D. Perkins, J. Schoenwaelder

RFC 2581
TCP Congestion Control M. Allman, V. Paxson, W. Stevens

RFC 2583
Guidelines for Next Hop Client (NHC) Developers R. Carlson, L. Winkler

RFC 2591
: *Definitions of Managed Objects for Scheduling Management Operations* D. Levi, J. Schoenwaelder

RFC 2625
: *IP and ARP over Fibre Channel* M. Rajagopal, R. Bhagwat, W. Rickard

RFC 2635
: *Don't SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam*)* S. Hambridge, A. Lunde

RFC 2637
: *Point-to-Point Tunneling Protocol* K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, G. Zorn

RFC 2640
: *Internationalization of the File Transfer Protocol* B. Curtin

RFC 2665
: *Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson

RFC 2671
: *Extension Mechanisms for DNS (EDNS0)* P. Vixie

RFC 2672
: *Non-Terminal DNS Name Redirection* M. Crawford

RFC 2675
: *IPv6 Jumbograms* D. Borman, S. Deering, R. Hinden

RFC 2710
: *Multicast Listener Discovery (MLD) for IPv6* S. Deering, W. Fenner, B. Haberman

RFC 2711
: *IPv6 Router Alert Option* C. Partridge, A. Jackson

RFC 2740
: *OSPF for IPv6* R. Coltun, D. Ferguson, J. Moy

RFC 2753
: *A Framework for Policy-based Admission Control* R. Yavatkar, D. Pendarakis, R. Guerin

RFC 2782
: *A DNS RR for specifying the location of services (DNS SRV)* A. Gubrandsen, P. Vixix, L. Esibov

RFC 2821
: *Simple Mail Transfer Protocol* J. Klensin, Ed.

RFC 2822
: *Internet Message Format* P. Resnick, Ed.

RFC 2840
: *TELNET KERMIT OPTION* J. Altman, F. da Cruz

RFC 2845
: *Secret Key Transaction Authentication for DNS (TSIG)* P. Vixie, O. Gudmundsson, D. Eastlake 3rd, B. Wellington

RFC 2851
: *Textual Conventions for Internet Network Addresses* M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder

RFC 2852
: *Deliver By SMTP Service Extension* D. Newman

RFC 2874
: *DNS Extensions to Support IPv6 Address Aggregation and Renumbering* M. Crawford, C. Huitema

RFC 2915
: *The Naming Authority Pointer (NAPTR) DNS Resource Record* M. Mealling, R. Daniel

RFC 2920
: *SMTP Service Extension for Command Pipelining* N. Freed

RFC 2930
: *Secret Key Establishment for DNS (TKEY RR)* D. Eastlake, 3rd

RFC 2941
: *Telnet Authentication Option* T. Ts'o, ed., J. Altman

RFC 2942
: *Telnet Authentication: Kerberos Version 5* T. Ts'o

RFC 2946
: *Telnet Data Encryption Option* T. Ts'o

RFC 2952
: *Telnet Encryption: DES 64 bit Cipher Feedback* T. Ts'o

RFC 2953
: *Telnet Encryption: DES 64 bit Output Feedback* T. Ts'o

RFC 2992
: *Analysis of an Equal-Cost Multi-Path Algorithm* C. Hopps

RFC 3019
: *IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol* B. Haberman, R. Worzella

RFC 3060
: *Policy Core Information Model—Version 1 Specification* B. Moore, E. Ellesson, J. Strassner, A. Westerinen

RFC 3152
: *Delegation of IP6.ARPA* R. Bush

RFC 3164
: *The BSD Syslog Protocol* C. Lonvick

RFC 3207
: *SMTP Service Extension for Secure SMTP over Transport Layer Security* P. Hoffman

RFC 3226
: *DNSSEC and IPv6 A6 aware server/resolver message size requirements* O. Gudmundsson

RFC 3291
: *Textual Conventions for Internet Network Addresses* M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder

RFC 3363
: *Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System* R. Bush, A. Durand, B. Fink, O. Gudmundsson, T. Hain

RFC 3376
Internet Group Management Protocol, Version 3 B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan

RFC 3390
Increasing TCP's Initial Window M. Allman, S. Floyd, C. Partridge

RFC 3410
Introduction and Applicability Statements for Internet-Standard Management Framework J. Case, R. Mundy, D. Partain, B. Stewart

RFC 3411
An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks D. Harrington, R. Presuhn, B. Wijnen

RFC 3412
Message Processing and Dispatching for the Simple Network Management Protocol (SNMP) J. Case, D. Harrington, R. Presuhn, B. Wijnen

RFC 3413
Simple Network Management Protocol (SNMP) Applications D. Levi, P. Meyer, B. Stewart

RFC 3414
User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) U. Blumenthal, B. Wijnen

RFC 3415
View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) B. Wijnen, R. Presuhn, K. McCloghrie

RFC 3416
Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP) R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 3417
Transport Mappings for the Simple Network Management Protocol (SNMP) R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 3418
Management Information Base (MIB) for the Simple Network Management Protocol (SNMP) R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 3419
Textual Conventions for Transport Addresses M. Daniele, J. Schoenwaelder

RFC 3484
Default Address Selection for Internet Protocol version 6 (IPv6) R. Draves

RFC 3493
Basic Socket Interface Extensions for IPv6 R. Gilligan, S. Thomson, J. Bound, J. McCann, W. Stevens

RFC 3513
Internet Protocol Version 6 (IPv6) Addressing Architecture R. Hinden, S. Deering

RFC 3526
More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) T. Kivinen, M. Kojo

RFC 3542
: *Advanced Sockets Application Programming Interface (API) for IPv6* W. Richard Stevens, M. Thomas, E. Nordmark, T. Jinmei

RFC 3566
: *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec* S. Frankel, H. Herbert

RFC 3569
: *An Overview of Source-Specific Multicast (SSM)* S. Bhattacharyya, Ed.

RFC 3584
: *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework* R. Frye, D. Levi, S. Routhier, B. Wijnen

RFC 3602
: *The AES-CBC Cipher Algorithm and Its Use with IPsec* S. Frankel, R. Glenn, S. Kelly

RFC 3629
: *UTF-8, a transformation format of ISO 10646* R. Kermode, C. Vicisano

RFC 3658
: *Delegation Signer (DS) Resource Record (RR)* O. Gudmundsson

RFC 3678
: *Socket Interface Extensions for Multicast Source Filters* D. Thaler, B. Fenner, B. Quinn

RFC 3715
: *IPsec-Network Address Translation (NAT) Compatibility Requirements* B. Aboba, W. Dixon

RFC 3810
: *Multicast Listener Discovery Version 2 (MLDv2) for IPv6* R. Vida, Ed., L. Costa, Ed.

| **RFC 3826**
| : *The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP
| User-based Security Model* U. Blumenthal, F. Maino, K McCloghrie.

RFC 3947
: *Negotiation of NAT-Traversal in the IKE* T. Kivinen, B. Swander, A. Huttunen, V. Volpe

RFC 3948
: *UDP Encapsulation of IPsec ESP Packets* A. Huttunen, B. Swander, V. Volpe, L. DiBurro, M. Stenberg

RFC 4001
: *Textual Conventions for Internet Network Addresses* M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder

RFC 4007
: *IPv6 Scoped Address Architecture* S. Deering, B. Haberman, T. Jinmei, E. Nordmark, B. Zill

RFC 4022
: *Management Information Base for the Transmission Control Protocol (TCP)* R. Raghunarayan

RFC 4106
: *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)* J. Viega, D. McGrew

RFC 4109
Algorithms for Internet Key Exchange version 1 (IKEv1) P. Hoffman

RFC 4113
Management Information Base for the User Datagram Protocol (UDP) B. Fenner, J. Flick

RFC 4191
Default Router Preferences and More-Specific Routes R. Draves, D. Thaler

RFC 4217
Securing FTP with TLS P. Ford-Hutchinson

RFC 4292
IP Forwarding Table MIB B. Haberman

RFC 4293
Management Information Base for the Internet Protocol (IP) S. Routhier

RFC 4301
Security Architecture for the Internet Protocol S. Kent, K. Seo

RFC 4302
IP Authentication Header S. Kent

RFC 4303
IP Encapsulating Security Payload (ESP) S. Kent

RFC 4304
Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP) S. Kent

RFC 4307
Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2) J. Schiller

RFC 4308
Cryptographic Suites for IPsec P. Hoffman

RFC 4434
The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol P. Hoffman

RFC 4443
Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification A. Conta, S. Deering

RFC 4552
Authentication/Confidentiality for OSPFv3 M. Gupta, N. Melam

RFC 4678
Server/Application State Protocol v1 A. Bivens

RFC 4753
ECP Groups for IKE and IKEv2 D. Fu, J. Solinas

RFC 4754
IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA) D. Fu, J. Solinas

RFC 4809
Requirements for an IPsec Certificate Management Profile C. Bonatti, Ed., S. Turner, Ed., G. Lebovitz, Ed.

RFC 4835
: *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)* V. Manral

RFC 4862
: *IPv6 Stateless Address Autoconfiguration* S. Thomson, T. Narten, T. Jinmei

RFC 4868
: *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec* S. Kelly, S. Frankel

RFC 4869
: *Suite B Cryptographic Suites for IPsec* L. Law, J. Solinas

RFC 4941
: *Privacy Extensions for Stateless Address Autoconfiguration in IPv6* T. Narten, R. Draves, S. Krishnan

RFC 4945
: *The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX* B. Korver

RFC 5014
: *IPv6 Socket API for Source Address Selection* E. Nordmark, S. Chakrabarti, J. Laganier

RFC 5095
: *Deprecation of Type 0 Routing Headers in IPv6* J. Abley, P. Savola, G. Neville-Neil

RFC 5175
: *IPv6 Router Advertisement Flags Option* B. Haberman, Ed., R. Hinden

RFC 5282
: *Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol* D. Black, D. McGrew

RFC 5996
: *Internet Key Exchange Protocol Version 2 (IKEv2)* C. Kaufman, P. Hoffman, Y. Nir, P. Eronen

Internet drafts

Internet drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Other groups can also distribute working documents as Internet drafts. You can see Internet drafts at http://www.ietf.org/ID.html.

Appendix D. Accessibility

Publications for this product are offered in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when using PDF files, you can view the information through the z/OS Internet Library website or IBM Knowledge Center. If you continue to experience problems, send an email to mhvrcfs@us.ibm.com or write to:

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. See z/OS TSO/E Primer, z/OS TSO/E User's Guide, and z/OS ISPF User's Guide Vol I for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer or Library Server versions of z/OS books in the Internet library at www.ibm.com/systems/z/os/zos/bkserv/.

One exception is command syntax that is published in railroad track format, which is accessible using screen readers with IBM Knowledge Center, as described in "Dotted decimal syntax diagrams."

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing IBM Knowledge Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should see separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- A question mark (?) means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- An exclamation mark (!) means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted

decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- An asterisk (*) means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

  Notes:

  1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
  2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
  3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.
- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Collection (SK3T-4269), which contains BookManager and PDF formats.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: http://www-01.ibm.com/software/support/systemsz/lifecycle/
- For information about currently-supported IBM hardware, contact your IBM representative.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Trademarks

Bibliography

This bibliography contains descriptions of the documents in the z/OS Communications Server library.

z/OS Communications Server documentation is available in the following forms:
- Online at the z/OS Internet Library web page at www.ibm.com/systems/z/os/zos/bkserv/
- In softcopy on CD-ROM collections. See "Softcopy information" on page xxxi.

z/OS Communications Server library updates

An index to z/OS Communications Server book updates is at http://www.ibm.com/support/docview.wss?uid=swg21178966. Updates to documents are also available on RETAIN® and in information APARs (info APARs). Go to http://www.ibm.com/software/network/commserver/zos/support to view information APARs.

z/OS Communications Server information

z/OS Communications Server product information is grouped by task in the following tables.

Planning

Title	Number	Description
z/OS Communications Server: New Function Summary	GC27-3664	This document is intended to help you plan for new IP or SNA function, whether you are migrating from a previous version or installing z/OS for the first time. It summarizes what is new in the release and identifies the suggested and required modifications needed to use the enhanced functions.
z/OS Communications Server: IPv6 Network and Application Design Guide	SC27-3663	This document is a high-level introduction to IPv6. It describes concepts of z/OS Communications Server's support of IPv6, coexistence with IPv4, and migration issues.

Resource definition, configuration, and tuning

Title	Number	Description
z/OS Communications Server: IP Configuration Guide	SC27-3650	This document describes the major concepts involved in understanding and configuring an IP network. Familiarity with the z/OS operating system, IP protocols, z/OS UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this document with the z/OS Communications Server: IP Configuration Reference.

Title	Number	Description
z/OS Communications Server: IP Configuration Reference	SC27-3651	This document presents information for people who want to administer and maintain IP. Use this document with the z/OS Communications Server: IP Configuration Guide. The information in this document includes: • TCP/IP configuration data sets • Configuration statements • Translation tables • Protocol number and port assignments
z/OS Communications Server: SNA Network Implementation Guide	SC27-3672	This document presents the major concepts involved in implementing an SNA network. Use this document with the z/OS Communications Server: SNA Resource Definition Reference.
z/OS Communications Server: SNA Resource Definition Reference	SC27-3675	This document describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. Use this document with the z/OS Communications Server: SNA Network Implementation Guide.
z/OS Communications Server: SNA Resource Definition Samples	SC27-3676	This document contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.
z/OS Communications Server: IP Network Print Facility	SC27-3658	This document is for systems programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP Services.

Operation

Title	Number	Description
z/OS Communications Server: IP User's Guide and Commands	SC27-3662	This document describes how to use TCP/IP applications. It contains requests with which a user can log on to a remote host using Telnet, transfer data sets using FTP, send and receive electronic mail, print on remote printers, and authenticate network users.
z/OS Communications Server: IP System Administrator's Commands	SC27-3661	This document describes the functions and commands helpful in configuring or monitoring your system. It contains system administrator's commands, such as TSO NETSTAT, PING, TRACERTE and their UNIX counterparts. It also includes TSO and MVS commands commonly used during the IP configuration process.
z/OS Communications Server: SNA Operation	SC27-3673	This document serves as a reference for programmers and operators requiring detailed information about specific operator commands.
z/OS Communications Server: Quick Reference	SC27-3665	This document contains essential information about SNA and IP commands.

Customization

Title	Number	Description
z/OS Communications Server: SNA Customization	SC27-3666	This document enables you to customize SNA, and includes the following information: • Communication network management (CNM) routing table • Logon-interpret routine requirements • Logon manager installation-wide exit routine for the CLU search exit • TSO/SNA installation-wide exit routines • SNA installation-wide exit routines

Writing application programs

Title	Number	Description
z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference	SC27-3660	This document describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this document to adapt your existing applications to communicate with each other using sockets over TCP/IP.
z/OS Communications Server: IP CICS Sockets Guide	SC27-3649	This document is for programmers who want to set up, write application programs for, and diagnose problems with the socket interface for CICS using z/OS TCP/IP.
z/OS Communications Server: IP IMS Sockets Guide	SC27-3653	This document is for programmers who want application programs that use the IMS™ TCP/IP application development services provided by the TCP/IP Services of IBM.
z/OS Communications Server: IP Programmer's Guide and Reference	SC27-3659	This document describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the z/OS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.
z/OS Communications Server: SNA Programming	SC27-3674	This document describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.
z/OS Communications Server: SNA Programmer's LU 6.2 Guide	SC27-3669	This document describes how to use the SNA LU 6.2 application programming interface for host application programs. This document applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this document.)
z/OS Communications Server: SNA Programmer's LU 6.2 Reference	SC27-3670	This document provides reference material for the SNA LU 6.2 programming interface for host application programs.
z/OS Communications Server: CSM Guide	SC27-3647	This document describes how applications use the communications storage manager.

Title	Number	Description
z/OS Communications Server: CMIP Services and Topology Agent Guide	SC27-3646	This document describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The document provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis

Title	Number	Description
z/OS Communications Server: IP Diagnosis Guide	GC27-3652	This document explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.
z/OS Communications Server: ACF/TAP Trace Analysis Handbook	GC27-3645	This document explains how to gather the trace data that is collected and stored in the host processor. It also explains how to use the Advanced Communications Function/Trace Analysis Program (ACF/TAP) service aid to produce reports for analyzing the trace data information.
z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures and z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT	GC27-3667 GC27-3668	These documents help you identify an SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation.
z/OS Communications Server: SNA Data Areas Volume 1 and z/OS Communications Server: SNA Data Areas Volume 2	GC31-6852 GC31-6853	These documents describe SNA data areas and can be used to read an SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and codes

Title	Number	Description
z/OS Communications Server: SNA Messages	SC27-3671	This document describes the ELM, IKT, IST, IUT, IVT, and USS messages. Other information in this document includes: • Command and RU types in SNA messages • Node and ID types in SNA messages • Supplemental message-related information
z/OS Communications Server: IP Messages Volume 1 (EZA)	SC27-3654	This volume contains TCP/IP messages beginning with EZA.
z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)	SC27-3655	This volume contains TCP/IP messages beginning with EZB or EZD.
z/OS Communications Server: IP Messages Volume 3 (EZY)	SC27-3656	This volume contains TCP/IP messages beginning with EZY.
z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)	SC27-3657	This volume contains TCP/IP messages beginning with EZZ and SNM.
z/OS Communications Server: IP and SNA Codes	SC27-3648	This document describes codes and other information that appear in z/OS Communications Server messages.

Index

Communicating your comments to IBM

If you especially like or dislike anything about this document, you can send us comments electronically by using one of the following methods:

Internet email:
comsvrcf@us.ibm.com

World Wide Web:
http://www.ibm.com/systems/z/os/zos/webqs.html

If you would like a reply, be sure to include your name, address, and telephone number. Make sure to include the following information in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Product Number: 5650-ZOS

Printed in USA

SC27-3651-03