z/OS

TSO/E System Programming Command Reference

Version 2 Release 1

SA32-0974-00

Note

Before using this information and the product it supports, read the information in "Notices" on page 125.

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

Contents

Figures

Tables

About this document

This publication describes the syntax of TSO/E commands that can be used by a system programmer to:

- Add, change, or delete entries in SYS1.UADS and the broadcast data set
- Review the contents of SYS1.UADS
- Synchronize the broadcast data set with either the RACF® data base, SYS1.UADS, or both
- Perform functions similar to that of the system operator, such as:
  - Cancelling a terminal session
  - Displaying information concerning system activity
  - Monitoring system activities
  - Sending messages
  - Using SLIP to trap and debug system errors
  - Terminating the monitoring activities
- Display, dynamically update, or check the syntax of the IKJTSOxx member of SYS1.PARMLIB
- Convert entries in SYS1.UADS to the RACF data base
- Test an authorized program or command processor
- Test installation-written exits and debug user-written control records
- Delete (remove from use through the virtual lookaside facility) partitioned data sets or named collections of data
- Establish an extended MCS console session and perform MVS™ operator activities
- Maintain a profile to tailor message processing during an extended MCS console session
- Debug system errors and/or CLIST errors
- Debug problems in the Information Center Facility CLISTs and REXX execs.

Note: When you see the term JESPLEX in this publication, understand it to mean either a logical grouping of JES2 systems that share the same multi-access spool (MAS) or a logical grouping of JES3 systems (each JES3 system consisting of one global JES3 system and some number of local JES3 systems).

Who should use this document

This book is intended for system programmers who install, maintain, and customize TSO/E.

How this document is organized

This book has three chapters as follows:

- Chapter 1, "Introduction," on page 1 describes the conventions and notations used for command syntax throughout the book. It also contains rules for coding commands and subcommands.
- Chapter 2, "Command syntax," on page 5 describes the syntax of individual TSO/E commands and subcommands.

- Chapter 3, “Information Center Facility trace commands,” on page 119 briefly describes the Information Center Facility trace commands that you can use to diagnose problems with the Facility's CLISTs and REXX execs.

How to use this document

If you are not familiar with TSO/E, read Chapter 1, “Introduction,” on page 1 to learn about the syntax conventions and coding rules for the system commands described in this book. Chapter 2, “Command syntax,” on page 5 describes the use of the individual commands and subcommands and is intended as a reference for system programming. Read Chapter 3, “Information Center Facility trace commands,” on page 119 if you need information on trace functions for the Information Center Facility's CLISTs and REXX execs.

Where to find more information

Please see *z/OS Information Roadmap* for an overview of the documentation associated with z/OS®, including the documentation available for z/OS TSO/E.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   US
4. Fax the comments to us, as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:
- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R1.0 TSO/E System Programming Command Reference
  SA32-0974-00
- The topic and page number that is related to your comment.
- The text of your comment.

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

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:
- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS support page (http://www.ibm.com/systems/z/support/).

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

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

Chapter 1. Introduction

This chapter illustrates the conventions and notations used to present the syntax of the commands and subcommands described in subsequent chapters. Also included are specific rules concerning the coding of a command or subcommand; the use of continuation lines and delimiters; and parameter definitions.

Syntax conventions and notations

The notation used to define the syntax in this publication is described in the following paragraphs.

1. The set of symbols listed below is used to define the syntax, but do not specify any of them in a command or subcommand.

   hyphen
   -

   underscore
   _

   braces { }

   brackets
   [ ]

   logical OR
   |

   ellipsis
   ...

   blank ƀ

   The special uses of the foregoing symbols are explained in the following paragraphs.

2. Specify uppercase letters, numbers, and the set of symbols shown in the following list exactly as shown in the syntax.

   single quotation mark
   '

   asterisk
   *

   comma
   ,

   equal sign
   =

   parentheses
   ( )

   percent
   %

   period .

   question mark
   ?

colon :

3. Lowercase letters, and symbols appearing in the syntax represent variables for which you substitute specific information.

   Example: If *name* appears in the syntax, substitute a specific value (for example, ALPHA) for the variable.
4. Hyphens join lower-case words and symbols to form a single variable

   Example: If *member-name* appears in the syntax, substitute a specific value (for example, BETA) for the variable.
5. A stack or the logical OR indicate related, alternative items. Select only **one** of the items (an *exclusive OR*); and specify it explicitly.

   Example: The representation

   ```
   A
   B    or    A|B|C
   C
   ```

   indicates select A or B or C and explicitly specify the selected item.
6. A stack or the logical OR also indicate related, alternative items, where one of the items is the default. An underscore indicates a default alternative. Select only **one** of the items (an *exclusive OR*). If you select an underscored alternative, you need not specify it explicitly. The absence of an explicit specification is an *implicit* specification of the default alternative.

   Example: The representation

   $$\begin{array}{lll} \texttt{A} & & \\ \underline{\texttt{B}} & \texttt{or} & \texttt{A}|\underline{\texttt{B}}|\texttt{C} \\ \texttt{C} & & \end{array}$$

   indicates select A or B or C; however, if you select B, you need not explicitly specify it because it is the default alternative.
7. To prevent ambiguity in syntax presentation, braces are used to group related, alternative items. Select only **one** of the items (an *exclusive OR*); and specify it explicitly.

   Example: The representation
   indicates choose one of the items enclosed within the braces. If you select A,

   $$\text{ALPHA} = \left(\left\{ \begin{array}{c} \text{A} \\ \text{B} \\ \text{C} \end{array} \right\},\ \text{D} \right) \qquad \text{or} \qquad \text{ALPHA} = (\ \{\ \text{A} \mid \text{B} \mid \text{C}\ \},\ \text{D})$$

   specify ALPHA=(A,D).
8. To prevent ambiguity in syntax presentation, braces are also used to group related alternative items, where one of the items is the default. An underscore indicates a default alternative. Select only **one** of the items (an *exclusive OR*). If you select an underscored alternative, you need not specify it explicitly. The absence of an explicit specification is an *implicit* specification of the default alternative.

   Example: The representation

$$\text{ALPHA} = \left(\left\{ \begin{matrix} \underline{\text{A}} \\ \text{B} \\ \text{C} \end{matrix} \right\}, \text{D} \right) \qquad \text{or} \qquad \text{ALPHA} = (\{ \underline{\text{A}} \mid \text{B} \mid \text{C} \}, \text{D})$$

indicates choose one of the items enclosed within the braces. If you select A, specify either ALPHA=(A,D) or ALPHA=(,D). If you select A, you need not specify it explicitly because it is the default alternative.

9. To indicate optionality and to prevent ambiguity in syntax presentation, brackets also group related, alternative items; however, everything within the brackets is optional. If you do not **explicitly** specify one of the items, the result is a **null** specification.

   Example: The representation
   indicates choose **one** of the items enclosed within the brackets (an *exclusive*

$$\text{ALPHA} = \left(\left[\begin{matrix} \text{A} \\ \text{B} \\ \text{C} \end{matrix} \right], \text{D} \right) \qquad \text{or} \qquad \text{ALPHA} = ([\text{A} \mid \text{B} \mid \text{C}], \text{D})$$

   OR) or omit all of the items within the brackets. If you select only D, specify ALPHA=(,D).

10. An ellipsis indicates that you may specify the preceding item or group of items:
    - zero times
    - once, or
    - more than once in succession

    Example:

    ```
    ALPHA[,BETA]...
    ```

    indicates that ALPHA can appear alone or can be followed by ,BETA any number of times in succession.

11. Alphanumeric characters: unless otherwise indicated, an alphanumeric character is one of the following:
    - alphabetic: A-Z
    - numeric: 0-9
    - special: $ # @

Continuation lines

Continue a command or subcommand, except the SLIP subcommand of OPERATOR, on one or more lines by following this rule:

Use either a hyphen (minus sign) or a plus sign as the last character on the line you wish to continue. If you use a plus sign, precede it by at least one blank to prevent the concatenation of character strings from line to line. (The plus sign causes TSO/E to delete leading delimiters (blanks, commas, tabs, and comments on the continuation line.)

Continue the SLIP subcommand of OPERATOR on one or more lines by following this rule:
- Use a blank as the last character on the line you wish to continue.

Continuation Lines

You can end a line of input anywhere except:
- An equal sign and its preceding keyword must appear on the same line.
- The binary indicator: (*b*) in the DATA keyword must appear on the same line.
- The complete keyword must appear on the same line.

Delimiters

For all subcommands, except the SLIP subcommand of OPERATOR, the following rule on the use of delimiters applies: unless otherwise indicated, use a blank (or blanks) as the delimiter between a subcommand and a following parameter, and between parameters.

For the SLIP subcommand of OPERATOR, the following rule applies: blanks are not allowed except between SLIP and SET, MOD, or DEL. For the other parameters of the SLIP subcommand of OPERATOR, a comma can be used as a delimiter.

Parameter definitions

Two types of parameters are described under the ACCOUNT and OPERATOR commands: positional and keyword. Code positional parameters exactly as shown in the syntax. Code keyword parameters in any order.

Chapter 2. Command syntax

This chapter describes the syntax of the following commands and subcommands:

- ACCOUNT command
  - Subcommands of ACCOUNT

    ADD Add new entries to the UADS and the broadcast data set; add new data to existing entries.

    CHANGE
    Change data in UADS entries; change user IDs in the broadcast data set.

    DELETE
    Delete entries or parts of entries from the UADS; delete user IDs from the broadcast data set.

    END Terminate the ACCOUNT command.

    HELP Display a list of the subcommands of the ACCOUNT command, along with the function, syntax, and parameters of the subcommands (not available for batch jobs).

    LIST Display the contents of an entry (or entries) in the UADS.

    LISTIDS
    Display the user IDs for all entries in the UADS.

    SYNC Synchronize the broadcast data set with either the RACF* data base, SYS1.UADS, or both.
- CONSOLE command
  - Establish an extended MCS console session and perform MVS operator activities
  - Subcommands of CONSOLE

    In conversational mode, the following subcommands can be used:

    CART Specify a command and response token (CART) to relate system commands (requests) with their corresponding responses.

    END Exit conversational mode.

    END DEACTIVATE
    End a console session.

    HELP Display a list of the valid CONSOLE subcommands or help information for specific subcommands.

    TSO Enter a TSO/E command.

    system-command
    Enter an MVS system or subsystem command.
- CONSPROF command
  - Maintain a profile to tailor message processing during an extended MCS console session.
- OPERATOR command
  - Subcommands of OPERATOR

CANCEL
: Cancel a terminal session.

DISPLAY
: Display summary or detailed information about users and jobs, the time of day and the date, summary or detailed information about SLIP traps, summary or detailed information about page and/or swap data sets, and summary or detailed information about the system-managed storage configuration.

END Terminate the OPERATOR command.

HELP Display a list of the subcommands of the OPERATOR command, along with the function, syntax, and parameters of the subcommands.

MONITOR
: Monitor both terminal and background job activities within the system. The system displays informational messages at your terminal.

SEND Send a message to other terminal users or operators.

SLIP Control SLIP/PER (serviceability level indication processing/program event recording).

STOPMN
: Terminate the monitoring operations of the MONITOR subcommand. The system terminates the display of informational messages at your terminal.

- PARMLIB command
  - maintain the active IKJTSOxx member of SYS1.PARMLIB, which includes listing the specifications and dynamically updating the member without a re-IPL. You can also check the syntax of any IKJTSOxx member of SYS1.PARMLIB.
- RACONVRT command
  - convert entries in SYS1.UADS to the RACF data base.
- RECEIVE command

  only selected parameters
  : You may use the selected parameters, in conjunction with those on the TRANSMIT command, to test your exit routines and debug user-written control records.
- SYNC command
  - synchronize the broadcast data set with the RACF data base, SYS1.UADS, or both.
- TESTAUTH command
  - test an authorized program or command processor.
- TRANSMIT command

  only selected parameters
  : You may use the selected parameters, in conjunction with those on the RECEIVE command, to test your exit routines and debug user-written control records.
- VLFNOTE command

  only selected parameters
  : You may use the selected parameters to delete (remove from use through VLF)
    - an entire class of IBM-supplied or user-supplied data.

– a named collection of data from an IBM-supplied class.
– partitioned data sets (PDSs) associated with a particular volume serial from an IBM-supplied or user-supplied class.

ACCOUNT command

Use the ACCOUNT command and its subcommands to manage the entries in the user attribute data set (SYS1.UADS) and in the broadcast data set. You can execute the command and subcommands in either a time-sharing session or as a batch job.

The UADS contains information about each terminal user who is authorized to use TSO/E. In turn, TSO/E uses that information to regulate access to the system.

The broadcast data set can contain notices and mail for the user IDs defined in SYS1.UADS.

The syntax of the ACCOUNT command is:

```
ACCOUNT
```

Allocate SYS1.UADS as SHR prior to using the ACCOUNT command. When issued from a batch environment, the ACCOUNT command and its subcommands access the broadcast data set via the SYSLBC DD name. If no SYSLBC DD statement is specified in the job, the active broadcast data set will be used by the command. If a SYSLBC DD statement is specified in the job, the referenced data set will be used as the broadcast data set.

The ADD, CHANGE, DELETE, and LIST subcommands contain a positional parameter that is designated as the **nodelist**. Each item in the **nodelist** (that is, *userid*, *password*, *acct-nmbr*, *proc*) is designated as a **node** in a hierarchical structure, with *userid* at the highest level and *proc* at the lowest level.

The syntax definitions of the ADD, CHANGE, and DELETE subcommands, are presented in two forms for each subcommand.

1. for ADD
   - **create mode** - add new TSO/E users, that is, create new entries in SYS1.UADS and broadcast data set
   - **add mode** - add data (nodes, attributes) to existing entries in SYS1.UADS
2. for CHANGE
   - change data (nodes, attributes) in entries in the UADS, and user IDs in the broadcast data set
   - change only the requirements of a procedure (or procedures)
3. for DELETE
   - delete particular node(s) from an entry (or entries) in the UADS
   - delete an entire entry or delete a node together with all lower-level associated nodes

ACCOUNT command return codes

Table 1. ACCOUNT command return codes

0	Processing successful.
12	Processing unsuccessful. An error message has been issued.

ACCOUNT—ADD subcommand - create mode

Use the ADD subcommand to create entries in the UADS for new users of TSO/E. As you create an entry in the UADS, TSO/E creates a corresponding entry (user ID) in the broadcast data set for that user. For each entry that you create, TSO/E builds a "typical" user profile in the user profile table (UPT) for that user.

When creating an entry in the UADS, you can also select the following attributes for the new user:

- the region size that the user can request at logon
- the ability to use the ACCOUNT command
- the ability to use the OPERATOR command
- the ability to use the SUBMIT, STATUS, CANCEL, and OUTPUT commands
- the ability to specify performance groups at logon
- the ability to issue dynamic allocation requests that result in the need for volume mounting
- a default destination (remote workstation) that the user may use to process SYSOUT data sets
- default output classes, job classes, and SYSOUT classes that the user may use to control job processing

The syntax of the ADD subcommand of ACCOUNT to **create** an entry in the UADS is:

```
                        ┌ password ┐  ┌ acct-nmbr ┐
A[ DD]  (userid         │          │  │           │ proc)
                        └    *     ┘  └     *     ┘

                   ┌ rgn-size ┐
          {SIZE(   │          │ ) }  [ UNIT (name) ]  [ USERDATA (data) ]
                   └ default  ┘

            ┌ ACCT   ┐  ┌ DEST(id)┐     ┌ JCL   ┐  ┌ MAXSIZE (region) ┐
            └ NOACCT ┘  └ NODEST  ┘     └ NOJCL ┘  └ NOLIM            ┘

              ┌ MOUNT   ┐  ┌ OPER   ┐  ┌ HOLD (class) ┐  ┌ JOBCLASS(j-class) ┐
              └ NOMOUNT ┘  └ NOOPER ┘  └ NOHOLD       ┘  └ NOJOBCLASS        ┘

                ┌ MSGCLASS(o-class) ┐   ┌ SYSOUT(s-class) ┐  ┌ RECOVER   ┐
                └ NOMSGCLASS        ┘   └ NOSYSOUT        ┘  └ NORECOVER ┘

                         ┌ PERFORM (perf - group [{ b } perf - group ] . . . ) ┐
                         │                       [{ , }              ]         │
                         └ NOPERFORM                                           ┘
```

- The first parameter (enclosed within parentheses) is the positional parameter (nodelist); all others are keyword parameters.
- When you create a new entry, an asterisk (*) indicates a null field; that is, you cannot later add a password and/or an account number or change the asterisk-specification to an actual password and/or account number under that *userid*.

userid
specifies the user ID for a new entry in the UADS and the broadcast data set.

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

If you specify a *userid* that already exists in broadcast data set, TSO/E deletes all messages for that *userid*; it does not delete the *userid*.

If you plan to send data through a JES network using the TRANSMIT/RECEIVE commands, *do not* define a user ID that:
- is the same as the name of an external writer
- is the same as the name of a node defined to JES
- begins with any of the following characters: R, RM, RMT

password
A password under the new user ID

value: 1-8 alphanumeric characters

* A null field

acct_nmbr
An account number under the new user ID

value: 1-40 EBCDIC characters excluding the following characters:
- blank
- comma
- semicolon
- apostrophe
- tab

* A null field

proc
A procedure name under the new user ID

value: 1-8 alphanumeric characters, beginning with an alphabetic or special character

SIZE
The assigned minimum region size for the new procedure if
- you do not specify the region size in the LOGON pre-prompt exit routine
- this user does not specify the SIZE parameter on the LOGON command

In addition, if the preceding two conditions are true and you do not specify SIZE or specify SIZE(0), TSO/E assigns a region size based on the value of the REGION= parameter on the EXEC statement in this user's logon procedure or a default size based on JES initialization parameters.

If you specify in SIZE a minimum region size that is larger than MAXSIZE for this user, TSO/E sets SIZE equal to MAXSIZE.

rgn_size
Number of 1024-byte units of virtual storage for this user's private area

value: an integer in the range 0-2096128

UNIT
Default specification of device name when this user allocates new data sets but does not specify a volume. (Data sets allocated via the catalog are an exception. See the ALLOCATE command in *z/OS TSO/E Command Reference*.)

Note: The default specification for UNIT in the UADS is related to the logon procedure selected in the foreground, not to the user ID. For jobs submitted in the background, TSO/E does *not* use the default specification.

name
The name of a device or group of devices (for example, SYSDA)

value: 1-8 alphanumeric characters

USERDATA
Optional data under this user ID. The 2-byte field in the UADS is a four-digit hexadecimal number that represents the contents of data. This optional data can be used by this user's programs.

data
The optional data

value: 4 EBCDIC characters (valid characters 0-9 and A-F)

ACCT
Allow this user to use the ACCOUNT command

NOACCT
Do not allow this user to use the ACCOUNT command

DEST
The default remote destination for processing dynamically allocated SYSOUT data sets. The establishment of this default eliminates the need for ROUTE statements in this user's submitted batch jobs. This user can override the default destination through the use of the ALLOCATE, FREE, and other commands.

Note: If a user submits a job at a node other than that specified in the DEST parameter, that user will not be able to view held output.

id The default destination (remote workstation)

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

NODEST
This user must explicitly route dynamically allocated SYSOUT data sets for processing

JCL
Allow this user to use the SUBMIT, STATUS, CANCEL, and OUTPUT commands

NOJCL
Do not allow this user to use the named commands

MAXSIZE
The maximum region size that this user may request at logon. If you do not specify MAXSIZE or specify MAXSIZE=0, TSO/E assumes NOLIM.

region
The number of 1024-byte units of virtual storage for the user's private area

value: an integer in the range 0-2096128

NOLIM
Do not restrict this user to a maximum region size at logon

MOUNT
Allow this user to issue dynamic allocation requests that result in the need for volume mounting

The volume request can be either explicit (for example, when this user issues the ALLOCATE command) or implicit (for example, when this user issues commands that cause the allocation of temporary space).

This user will sit in a "locked out" condition at the terminal until the operator responds to the request. Therefore, this user should send a message to the operator prior to issuing the command requesting a particular volume.

NOMOUNT
Ignore the issuance by this user of dynamic allocation requests that result in the need for volume mounting

OPER
Allow this user to use the OPERATOR command

NOOPER
Do not allow this user to use the OPERATOR command

HOLD
Place the job output from a job submitted with the HOLD keyword on the SUBMIT command in the hold queue

class
Assign a default held output class to the submitted job

value: 1 alphanumeric character, excluding special characters

NOHOLD
Assign a default output class to the submitted job as indicated by the MSGCLASS parameter

JOBCLASS
Assign a default job class to a job submitted without a JOB statement

j_class
The default job class

value: 1 alphanumeric character, excluding special characters

NOJOBCLASS
JES2 or JES3 assigns the default job class to a job submitted without a JOB statement

MSGCLASS
If this user submits a job without a JOB statement or without the specification of the MSGCLASS= parameter on the JOB statement; and specifies the NOHOLD keyword on the SUBMIT command, assign the job to a default output class.

o_class
The default output class

value: 1 alphanumeric character, excluding special characters

NOMSGCLASS
If this user submits a job without a JOB statement or without the specification of the MSGCLASS= parameter on the JOB statement; and specifies the NOHOLD keyword on the SUBMIT command, JES2 or JES3 assigns the default output class

SYSOUT
If this user does not specify *class* with the SYSOUT keyword on an ALLOCATE command, assign a default SYSOUT class

s_class
The default SYSOUT class

value: 1 alphanumeric character, excluding special characters

NOSYSOUT
If this user does not specify *class* with the SYSOUT keyword on an ALLOCATE command, JES2 or JES3 assigns the default SYSOUT class

RECOVER
Allow this user to use the EDIT recovery facility

NORECOVER
Do not allow this user to use the EDIT recovery facility

PERFORM
Allow this user to explicitly request a performance group (or groups) at logon

perf_group
The identification of the performance group (or groups)

value: an integer in the range 1-255

Note: If you write an installation control specification for TSO/E users, the following applies to the interpretation of the PERFORM parameter under the stated conditions:

- TSO/E bypasses the performance group(s) in the UADS. When this user logs on, the performance group TSO/E assigns depends upon the values you specify for PGN and OPGN in the installation control specification and the value this user specifies at logon:
  - logon PERFORM value = OPGN value: accept logon value
  - logon PERFORM value ¬= OPGN value: assign PGN value
  - logon PERFORM value = PGN value: accept logon value
  - logon PERFORM value ¬= PGN value (no OPGN value specified): assign PGN value

NOPERFORM
Do not allow this user to request a performance group

ACCOUNT—ADD subcommand - add mode

Use the ADD subcommand also to add nodes to existing entries in the UADS. Do not use ADD to change any existing nodes in a UADS entry; use the CHANGE subcommand instead.

When making additions to the UADS, ADD ensures that no duplications will exist in the UADS structure. If you attempt to add a node that already exists at the specified location in an entry, no addition takes place.

The syntax of the ADD subcommand of ACCOUNT to **add** nodes to an existing entry is:

```
             ┌userid┐   ┌ password  ┌ acct-nmbr ┐┐
A[ DD]  (   {       }   [           [           ]]  )
             └  *   ┘   └    *      └    *      ┘┘

        ┌                                            ┐┌                                                ┐
        │ (password₁ ┌┌ ƀ ┐                 ┐       ││   (acct-nmbr₁ ┌┌ ƀ ┐                 ┐       │{                                  }
DATA(   │            [{    }  passwordₙ     ] . . . ) ││               [{    }  acct-nmbrₙ    ] . . . )│{ (proc₁ [{ ƀ }  procₙ ] . . . )    } )
        │            └└ , ┘                 ┘       ││               └└ , ┘                 ┘       │{        [{ , }        ]            }
        │                                            ││                                                │
        │                  *                         ││                      *                         │
        └                                            ┘└                                                ┘

          ┌ rgn-size ┐
   {SIZE( {          } )}  [UNIT (name)]
          └ default  ┘
```

- The first parameter (enclosed within parentheses) is a positional parameter (nodelist); all others are keyword parameters.
- If you created a new entry with an asterisk (*) to indicate a null field (that is, TSO/E does not support passwords and/or account numbers under that *userid*, any subsequent explicit specification in either the positional or DATA parameter of *password* and/or *acct-nmbr* for that *userid* is not valid.
- Specify in the DATA parameter all items you omit from the positional parameter.
- When you specify a list of passwords and/or account numbers and procedure names, separate each item in the list by a comma or a blank and enclose *each* list within a separate set of parentheses embedded within the set required for the DATA parameter. If you specify only a list of procedure names, the embedded parentheses are optional.

userid
: Add node(s) to an existing entry in the UADS.

  value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

`*` Add node(s) to all existing entries in the UADS

password
: Add node(s) (account number(s)/procedure(s)) under the password in an existing entry (or entries)

  value: 1-8 alphanumeric characters

`*` Add node(s) under all passwords in an existing entry (or entries)

acct_nmbr
: Add node(s) (procedure(s)) under the account number in an existing entry (or entries)

  value: 1-40 EBCDIC characters excluding the following characters:
  - blank
  - comma
  - semicolon
  - apostrophe
  - tab

`*` Add procedures under all account numbers in an existing entry (or entries).

DATA
: Add node(s) to an existing entry (or entries) in the UADS

  password
  : Add a password or a list of passwords

value: 1-8 alphanumeric characters

`*` No added passwords

acct_nmbr

Add an account number or a list of account numbers. Do not specify more than 255 identical account numbers under any one user ID.

value: 1-40 EBCDIC characters excluding the following characters:
- blank
- comma
- semicolon
- apostrophe
- tab

`*` No added account numbers

proc

Add a procedure name or a list of procedure names. Do not specify more than 255 identical procedure names under any one user ID.

value: 1-8 alphanumeric characters, beginning with an alphabetic or special character

SIZE

The assigned minimum region size for the new procedure if
- you do not specify the region size in the LOGON pre-prompt exit routine
- this user does not specify the SIZE parameter on the LOGON command

In addition, if the preceding two conditions are true and you do not specify SIZE or specify SIZE(0), TSO/E assigns a region size based on the value of the REGION= parameter on the EXEC statement in the user's logon procedure or a default size based on JES initialization parameters.

If you specify in SIZE a minimum region size that is larger than MAXSIZE for this user ID, TSO/E sets SIZE equal to MAXSIZE.

You can specify a SIZE parameter for each unique combination of password, account number, procedure name under a user ID.

rgn_size

Number of 1024-byte units of virtual storage for this user's private area

value: an integer in the range 0-2096128

UNIT

Default specification of device type when this user allocates new data sets but does not specify a volume. (Data sets allocated via the catalog are an exception. See the ALLOCATE command in *z/OS TSO/E Command Reference*.)

Note: The default specification for UNIT in the UADS is related to the logon procedure selected in the foreground, not to the user ID. For jobs submitted in the background, TSO/E does *not* use the default specification.

You can specify a UNIT parameter for each unique combination of password, account number, procedure name under a user ID.

name

The name of a device or group of devices (for example, SYSDA)

value: 1-8 alphanumeric characters

Example 1

Operation: Add a new entry to the UADS and the broadcast data set.

```
add (warner1 xaybzc 32058 mylog) noacct nooper jcl -
maxsize(150) size(80) unit(sysda) userdata(1fa0) -
dest(deptout) mount perform(1,5,6,2,4)
```

Example 2

Operation: Add a new password, account number, and procedure name to an existing entry in the UADS. Also include the region size requirements for the procedure.

```
add (warner1) data(mz3tii 7116166 amabala) size(20)
```

Example 3

Operation: Continuing Example 2, add a new account number and procedure name to an existing entry in the UADS.

```
add (warner1 mz3tii) data(288104 mylog) size(114) unit(sysda)
```

Example 4

Operation: Add a new procedure name, and the region size requirements for it, to all entries in the UADS.

```
add (* * *) data(mcqlg) size (73)
```

Example 5

Operation: Add a new account number and procedure name to all structures under an existing entry in the UADS.

```
add (warner1 *) data(5707571 logproc) size(100)
```

ACCOUNT—CHANGE subcommand

Use the CHANGE subcommand to change data (nodes and user attributes) in entries in the UADS and user IDs in the broadcast data set; or to change only the requirements of procedures.

When making changes to the UADS, CHANGE ensures that no identical (redundant) paths will exist in the UADS structure after the change operation. On the other hand, if an 'impossible merge' situation arises (identical procedure names associated with different data), CHANGE cannot determine which data to retain. Therefore, it terminates processing of the current structure and issues an explanatory message.

The syntax of the CHANGE subcommand of ACCOUNT to change nodes and user attributes within entries in the UADS and user IDs in the broadcast data set is:

```
                ⎧userid⎫⎡password  ⎡acct-nmbr ⎡proc ⎤⎤⎤
C[ HANGE]  (    ⎨      ⎬⎢          ⎢          ⎢     ⎥⎥⎥ )
                ⎩  *   ⎭⎣   *      ⎣    *     ⎣  *  ⎦⎦⎦

               ⎧userid   ⎫
               ⎪password ⎪
          DATA(⎨         ⎬)
               ⎪acct-nmbr⎪
               ⎩proc     ⎭

              ⎧rgn-size⎫
         {SIZE(⎨        ⎬) }  [ UNIT (name) ]   [ USERDATA(data) ]
              ⎩default ⎭

             ⎡ACCT  ⎤⎡DEST(id)⎤⎡JCL  ⎤⎡MAXSIZE(region)⎤
             ⎣NOACCT⎦⎣NODEST  ⎦⎣NOJCL⎦⎣NOLIM          ⎦

               ⎡MOUNT  ⎤⎡OPER  ⎤⎡HOLD(class)⎤⎡JOBCLASS(j-class)⎤
               ⎣NOMOUNT⎦⎣NOOPER⎦⎣NOHOLD     ⎦⎣NOJOBCLASS       ⎦

                 ⎡MSGCLASS(o-class)⎤⎡SYSOUT(s-class)⎤⎡RECOVER  ⎤
                 ⎣NOMSGCLASS       ⎦⎣NOSYSOUT       ⎦⎣NORECOVER⎦

                    ⎡                    ⎡⎧b̸⎫           ⎤       ⎤
                    ⎢PERFORM(perf-group  ⎢⎨  ⎬perf-group ⎥. . . )⎥
                    ⎢                    ⎣⎩,⎭           ⎦       ⎥
                    ⎣NOPERFORM                                  ⎦
```

- The first parameter (enclosed within parentheses) is the positional parameter (nodelist); all others are keyword parameters.
- To change a user ID in the UADS and the broadcast data set, explicitly specify *userid* as the only item in the positional parameter and specify *userid* in the DATA parameter. (The specification `c (*) data (`*`userid`*`)` is not valid.)
- To change a password, account number, or procedure name under a user ID (or user IDs), specify the item as the final item in the positional parameter, and specify the corresponding item in the DATA parameter.
- If you created (with the ADD subcommand) an entry in the UADS with an asterisk (*) specification for password and/or account number, you cannot change the asterisk-specification to an actual password and/or account number under that *userid*.
- If you change MAXSIZE, the specification of a region size smaller than an existing SIZE for any procedure under a user ID is not valid.
- The specification of SIZE and/or UNIT is valid only if you specify *proc* in the DATA parameter.

`userid`

Change either this user ID in the UADS and the broadcast data set, or data under this user ID

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

If you plan to send data through a JES2 network using the TRANSMIT/RECEIVE commands, *do not* define a user ID that:
- is the same as the name of an external writer
- or begins with any of the following characters: R, RM, RMT.

In the broadcast data set, the effect of `c` *userid*$_1$ `data (`*userid*$_2$`)` is as follows:

- If *userid*$_1$ and *userid*$_2$ do not exist, add *userid*$_2$ as a new entry.
- If *userid*$_1$ exists and *userid*$_2$ does not, delete *userid*$_1$; add *userid*$_2$ as a new entry; chain messages for *userid*$_1$ to *userid*$_2$.
- If *userid*$_1$ and *userid*$_2$ both exist, delete *userid*$_1$; chain messages for *userid*$_1$ to *userid*$_2$.

`*` Change data under all userids

`password`
Change either this password or node(s) under this password

value: 1-8 alphanumeric characters

`*` Change node(s) under all passwords

`acct_nmbr`
Change either this account number or node(s) under this account number

value: 1-40 EBCDIC characters excluding the following characters:
- blank
- comma
- semicolon
- apostrophe
- tab

`*` Change node(s) under all account numbers

`proc`
Change the procedure name

value: 1-8 alphanumeric characters, beginning with an alphabetic or special character

`*` Change all procedure names

`DATA`
Change specific nodes in an existing entry (or entries)

`userid`
The new user ID

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

`password`
The new password

value: 1-8 alphanumeric characters

`acct_nmbr`
The new account number

value: 1-40 EBCDIC characters excluding the following characters:
- blank
- comma
- semicolon
- apostrophe
- tab

`proc`
The new procedure name

value: 1-8 alphanumeric characters, beginning with an alphabetic or special character

SIZE
The assigned minimum region size for the new procedure if
- you do not specify the region size in the LOGON pre-prompt exit routine
- this user does not specify the SIZE parameter on the LOGON command

In addition, if the preceding two conditions are true and you do not specify SIZE or specify SIZE(0), TSO/E assigns a region size based on the value of the REGION= parameter on the EXEC statement in the user's logon procedure or a default size based on JES initialization parameters.

If you specify in SIZE a minimum region size that is larger than MAXSIZE for this user ID, TSO/E sets SIZE equal to MAXSIZE.

rgn_size
Number of 1024-byte units of virtual storage for this user's private area

value: an integer in the range 0-2096128

UNIT
Default specification of device type when this user allocates new data sets but does not specify a volume. (Data sets allocated using the catalog are an exception. See the ALLOCATE command in *z/OS TSO/E Command Reference*.)

Note: The default specification for UNIT in the UADS is related to the logon procedure selected in the foreground, not to the user ID. For jobs submitted in the background, TSO/E does *not* use the default specification.

name
The new name of a device or group of devices (for example, SYSDA)

value: 1-8 alphanumeric characters

USERDATA
Change optional data under this user ID. The 2-byte field in the UADS is a four-digit hexadecimal number that represents the contents of data. This hexadecimal data can be used by this user's programs.

data
The new optional data

value: 4 EBCDIC characters (valid characters 0-9 and A-F)

ACCT
Allow this user to use the ACCOUNT command

NOACCT
Do not allow this user to use the ACCOUNT command

DEST
The default remote destination for processing dynamically allocated SYSOUT data sets. The establishment of this default eliminates the need for ROUTE statements in this user's submitted batch jobs. The user can override the default destination through the use of the ALLOCATE, FREE, and other commands.

Note: If a user submits a job at a node other than that specified in the DEST parameter, that user will not be able to view held output.

id The default destination (remote workstation)

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

NODEST
This user must explicitly route SYSOUT data sets for processing

JCL
Allow this user to use the SUBMIT, STATUS, CANCEL, and OUTPUT commands

NOJCL
Do not allow this user to use the named commands

MAXSIZE
The maximum region size that this user may request at logon. If you do not specify MAXSIZE or specify MAXSIZE=0, TSO/E assumes NOLIM.

region
The number of 1024-byte units of virtual storage for this user's private area

value: an integer in the range 0-2096128

NOLIM
Do not restrict this user to a maximum region size at logon

MOUNT
Allow this user to issue dynamic allocation requests that result in the need for volume mounting

The volume request can be either explicit (for example, when this user issues the ALLOCATE command) or implicit (for example, when this user issues commands that cause the allocation of temporary space).

This user will sit in a “locked out” condition at the terminal until the operator responds to the request. Therefore, this user should send a message to the operator prior to issuing the command requesting a particular volume.

NOMOUNT
Ignore the issuance by this user of dynamic allocation requests that result in the need for volume mounting

OPER
Allow this user to use the OPERATOR command

NOOPER
Do not allow this user to use the OPERATOR command

HOLD
Place the job output from a job submitted with the HOLD keyword on the SUBMIT command in the hold queue

class
Assign a default held output class to the submitted job

value: 1 alphanumeric character, excluding special characters

NOHOLD
Assign a default output class to the submitted job as indicated by the MSGCLASS parameter

JOBCLASS
Assign a default job class to a job submitted without a JOB statement

j_class
The default job class

value: 1 alphanumeric character, excluding special characters

NOJOBCLASS
JES2 or JES3 assigns the default job class to a job submitted without a JOB statement

MSGCLASS
If this user submits a job without a JOB statement or without the specification of the MSGCLASS= parameter on the JOB statement; and specifies the NOHOLD keyword on the SUBMIT command, assign the job to a default output class.

o_class
The default output class

value: 1 alphanumeric character, excluding special characters

NOMSGCLASS
If this user submits a job without a JOB statement or without the specification of the MSGCLASS= parameter on the JOB statement; and specifies the NOHOLD keyword on the SUBMIT command, JES2 or JES3 assigns the default output class

SYSOUT
If this user does not specify *class* with the SYSOUT keyword on an ALLOCATE command, assign a default SYSOUT class

s_class
The default SYSOUT class

value: 1 alphanumeric character, excluding special characters

NOSYSOUT
If this user does not specify *class* with the SYSOUT keyword on an ALLOCATE command, JES2 or JES3 assigns the default SYSOUT class

RECOVER
Allow this user to use the EDIT recovery facility

NORECOVER
Do not allow this user to use the EDIT recovery facility

PERFORM
Allow this user to explicitly request a performance group (or groups) at logon

perf_group
The identification of the performance group (or groups)

value: an integer in the range 1-255

Note: If you write an installation control specification for TSO/E users, the following applies to the interpretation of the PERFORM parameter under the stated conditions:

- TSO/E bypasses the performance group(s) in the UADS. When a user logs on, the performance group TSO/E assigns depends upon the values you specify for PGN and OPGN in the installation control specification and the value the user specifies at logon:
  - logon PERFORM value = OPGN value: accept logon value
  - logon PERFORM value ¬= OPGN value: assign PGN value
  - logon PERFORM value = PGN value: accept logon value
  - logon PERFORM value ¬= PGN value (no OPGN value specified): assign PGN value

NOPERFORM
Do not allow this user to request a performance group

The syntax of the CHANGE subcommand of ACCOUNT to change only the requirements of a procedure (or procedures) is:

```
            ┌        ┐ ┌          ┐ ┌           ┐ ┌      ┐
            │ userid │ │ password │ │ acct-nmbr │ │ proc │
C[ HANGE]  (│        │ │          │ │           │ │      │
            │   *    │ │    *     │ │     *     │ │  *   │
            └        ┘ └          ┘ └           ┘ └      ┘

             ┌                                       ┐
             │        ┌          ┐                   │
             │        │ rgn-size │                   │
             │ SIZE(  │          │ )  [ UNIT(name) ] │
             │        │ default  │                   │
             │        └          ┘                   │
             │ UNIT(name)                            │
             └                                       ┘
```

- The first parameter (enclosed within parentheses) is the positional parameter (nodelist); all others are keyword parameters.

userid
Change the requirements of a procedure (or procedures) under this user ID

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

* Change the requirements of a procedure (or procedures) under all user IDs

password
Change the requirements of a procedure (or procedures) under this password

value: 1-8 alphanumeric characters

* Change the requirements of a procedure (or procedures) under all passwords

acct_nmbr
Change the requirements of a procedure (or procedures) under this account number

value: 1-40 EBCDIC characters excluding the following characters:
- blank
- comma
- semicolon
- apostrophe
- tab

* Change the requirements of a procedure (or procedures) under all account numbers

proc
Change the requirements of this procedure

value: 1-8 alphanumeric characters, beginning with an alphabetic or special character

* Change the requirements of all procedures under the specified nodes

SIZE
The assigned minimum region size for the new procedure(s) if
- you do not specify the region size in the LOGON pre-prompt exit routine
- this user does not specify the SIZE parameter on the LOGON command

In addition, if the preceding two conditions are true and you do not specify SIZE or specify SIZE(0), TSO/E assigns a region size based on the value of the REGION= parameter on the EXEC statement in the user's logon procedure or a default size based on JES initialization parameters.

If you specify in SIZE a minimum region size that is larger than MAXSIZE for this user, TSO/E sets SIZE equal to MAXSIZE.

`rgn_size`
: Number of 1024-byte units of virtual storage for this user's private area

  value: an integer in the range 0-2096128

`UNIT`

Change the default specification of device type when this user allocates data sets but does not specify a volume. (Data sets allocated via the catalog are an exception. See the ALLOCATE command in *z/OS TSO/E Command Reference*.)

Note: The default specification for UNIT in the UADS is related to the logon procedure selected in the foreground, not to the user ID. For jobs submitted in the background, TSO/E does *not* use the default specification.

`name`
: The name of a device or group of devices (for example, SYSDA)

  value: 1-8 alphanumeric characters

Example 1

Operation: Change an account number for an entry in the UADS and allow the user to issue the ACCOUNT and OPERATOR commands.

```
change (slc05 aox3p se29705) data(2e26705) acct oper
```

Example 2

Operation: Allow all users to issue the SUBMIT, CANCEL, STATUS, and OUTPUT commands.

```
change (*) jcl
```

The asterisk in the first positional parameter position specifies that all user IDs are considered valid for the operation of this subcommand.

Example 3

Operation: Change the user identification for an entry in the UADS.

```
change (warner) data(renwar)
```

Example 4

Operation: Change the name of a procedure in all nodes under user ID ja195.

```
change (ja195 * * oldproc) data(newproc)
```

Example 5

Operation: Change the default destination for an entry in the UADS.

```
change (ceh01) dest(rmt1)
```

ACCOUNT—DELETE subcommand

Use the DELETE subcommand to delete node(s) from the UADS and user IDs from the broadcast data set. Each terminal user has an entry in the UADS; and each entry may contain several nodes. The node(s) that you want to delete may be a part of an entry, or may be an entire entry.

The syntax of the DELETE subcommand of ACCOUNT to delete particular node(s) from an entry in the UADS is:

```
               ⎧ userid ⎫ ⎡ password  ⎡ acct-nmbr ⎤⎤
D[ ELETE]  (   ⎨        ⎬ ⎢           ⎢           ⎥⎥ )
               ⎩   *    ⎭ ⎣    *      ⎣     *     ⎦⎦

                    ⎧              ⎡⎧ b̸ ⎫              ⎤    ⎫
                    ⎪ password1    ⎢⎨   ⎬ passwordn    ⎥... ⎪
                    ⎪              ⎣⎩ , ⎭              ⎦    ⎪
                    ⎪                                       ⎪
                    ⎪              ⎡⎧ b̸ ⎫              ⎤    ⎪
              DATA( ⎨ acct-nmbr1   ⎢⎨   ⎬ acct-nmbrn   ⎥... ⎬ )
                    ⎪              ⎣⎩ , ⎭              ⎦    ⎪
                    ⎪                                       ⎪
                    ⎪          ⎡⎧ b̸ ⎫        ⎤              ⎪
                    ⎪ proc1    ⎢⎨   ⎬ procn  ⎥...           ⎪
                    ⎩          ⎣⎩ , ⎭        ⎦              ⎭
```

- The first parameter (enclosed within parentheses) is the positional parameter (nodelist); all others are keyword parameters.
- When you delete all nodes under a user ID, TSO/E deletes the user ID.
- The explicit specification of the same item in both the positional and DATA parameters is not valid.
- If you created (with the ADD subcommand) an entry in the UADS with an asterisk (*) specification for password and/or account number, an explicit specification of *password* and/or *acct-nmbr* for the particular user ID is not valid.

userid
Delete node(s) under this user ID

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

***** Delete node(s) under all user IDs

password
Delete node(s) under this password

value: 1-8 alphanumeric characters

***** Delete node(s) under all passwords

acct_nmbr
Delete node(s) under this account number

value: 1-40 EBCDIC characters excluding the following characters:
- blank
- comma
- semicolon
- apostrophe
- tab

`*` Delete node(s) under all account numbers

DATA
: Delete node(s) from an entry in the UADS

password
: Delete this password or the list of passwords and all node(s) under the password(s)

  value: 1-8 alphanumeric characters

acct_nmbr
: Delete this account number or the list of account numbers and all node(s) under the account number(s)

  value: 1-40 EBCDIC characters excluding the following characters:
  - blank
  - comma
  - semicolon
  - apostrophe
  - tab

proc
: Delete this procedure or the list of procedures

  value: 1-8 alphanumeric characters, beginning with an alphabetic or special character

The syntax of the DELETE subcommand of ACCOUNT to delete an entire entry from the UADS, or to delete a node and all lower-level associated nodes from an entry in the UADS is:

```
            ⎧ userid ⎫ ⎡ password ⎡ acct-nmbr ⎤⎤
D[ELETE]   (⎨        ⎬ ⎢          ⎢           ⎥⎥        )
            ⎩   *    ⎭ ⎣    *     ⎣     *     ⎦⎦
```

- The parameter enclosed within parentheses is positional (nodelist).
- To delete an entire entry in the UADS and a user ID (with associated messages) in the broadcast data set, specify:

  d (*userid*)
- To delete all entries in the UADS and all user IDs (and all messages) in the broadcast data set, specify:

  d (*)
- When you delete all nodes under a user ID, TSO/E deletes the user ID.
- If you created (with the ADD subcommand) an entry in the UADS with an asterisk (*) specification for password and/or account number, an explicit specification of *password* and/or *acct-nmbr* for the particular user ID is not valid.

userid
: Delete the entry from the UADS and the user ID from the broadcast data set, or delete node(s) under this user ID

  value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

`*` Delete all the entries from the UADS, and all user IDs and messages from the broadcast data set; or delete node(s) under all user IDs

password
Delete this password and all node(s) under this password, or delete node(s) under this password

value: 1-8 alphanumeric characters

`*` Delete node(s) under all passwords

acct_nmbr
Delete this account number and all node(s) under this account number

value: 1-40 EBCDIC characters excluding the following characters:
- blank
- comma
- semicolon
- apostrophe
- tab

`*` Delete all nodes under all account numbers

Example 1

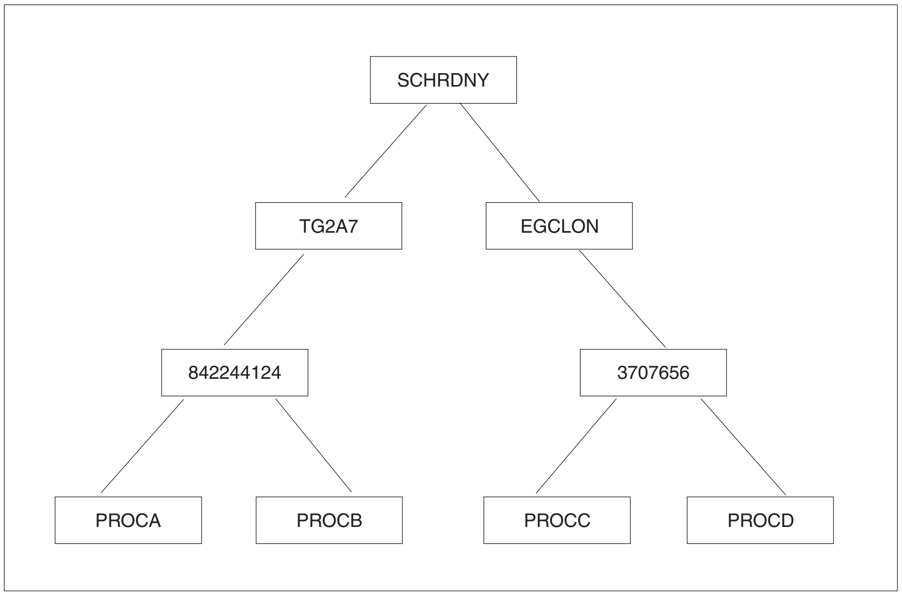
Operation: Delete an entire entry from the UADS.

```
delete (early08)
```

Example 2

Operation: Delete a procedure name from an entry in the UADS having the following structure:

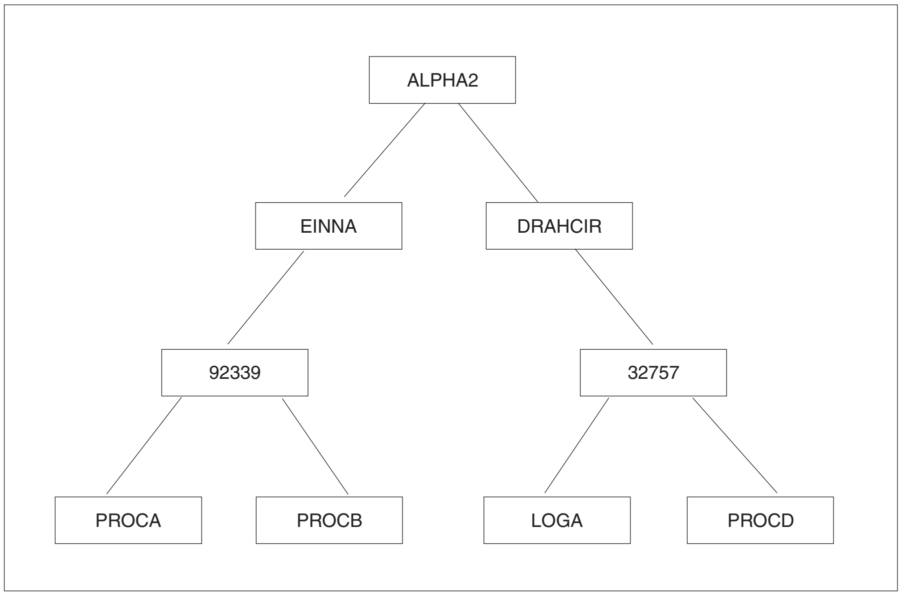
```
delete (schrdny egclon 3707656) data(procd)
```



Example 3

Operation: Delete an account number and all procedure names under that account number from an entry in the UADS having the following structure:

```
delete (alpha2 drahcir 32757)
```



ACCOUNT—END subcommand

Use the END subcommand to terminate operation of the ACCOUNT command.

The syntax of the END subcommand of ACCOUNT is:

```
END
```

ACCOUNT—HELP subcommand

Use the HELP subcommand to find out how to use the ACCOUNT subcommands. When you enter the HELP subcommand, TSO/E responds by displaying explanatory information at your terminal. You may request:

- A list of available subcommands
- An explanation of the function, syntax, and parameters (both positional and keyword) of a specific subcommand

The HELP subcommand actually causes TSO/E to execute a function of the HELP command; therefore, see the discussion of the HELP command in *z/OS TSO/E Command Reference*.

The syntax of the HELP subcommand of ACCOUNT is:

```
H[ ELP] [ subcmd-name [ ALL
                        FUNCTION
                        SYNTAX

                        OPERANDS[ (parm1 [{b̸ | ,} parmn]...) ] ] ]
```

- If you specify HELP with no parameters, TSO/E displays a list of available subcommands of ACCOUNT at your terminal.

subcmd_name
: The subcommand you want clarified

value: any valid subcommand of ACCOUNT

ALL
Display a description of the function, syntax, positional parameters, and keyword parameters of the subcommand

FUNCTION
Display a description of the function of the subcommand

SYNTAX
Display a description of the proper syntax of the subcommand

OPERANDS
Display a description of the positional and keyword parameters of the subcommand

parm
Display only a description of the indicated keyword parameter(s) of the subcommand

value: any valid keyword parameter of the subcommand

Example 1

Operation: Have a list of available subcommands displayed at your terminal.

```
help
```

Example 2

Operation: Obtain all available information about the ADD subcommand.

```
h add
```

Example 3

Operation: Have a list of the positional and keyword parameters for the CHANGE subcommand displayed at your terminal.

```
h change operands
```

Example 4

Operation: Have a list of the indicated keyword parameters for the ADD subcommand displayed at your terminal.

```
h add operands (data mount userdata)
```

ACCOUNT—LIST subcommand

Use the LIST subcommand to display entries in the UADS or to display nodes in particular entries.

The syntax of the LIST subcommand of ACCOUNT is:

```
              { userid } [ password [ acct-nmbr [ proc ]]]
L[ IST]  (    {        } [          [           [      ]]]  )
              {   *    } [    *     [     *     [  *   ]]]
```

• The parameter enclosed within parentheses is positional (nodelist).

userid
Display either all the nodes under this user ID, or some node(s) under this user ID

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

`*` Display either the contents of the UADS, or nodes under all user IDs

password
Display node(s) under this password

value: 1-8 alphanumeric characters

`*` Display the nodes under all passwords

acct_nmbr
Display node(s) under this account number

value: 1-40 EBCDIC characters excluding the following characters:
- blank
- comma
- semicolon
- apostrophe
- tab

`*` Display the nodes under all account numbers

proc
Display reference(s) to this procedure

value: 1-8 alphanumeric characters, beginning with an alphabetic or special character

`*` Display references to all procedures

Implicit specifications:
- The specification of L (*userid*) implies L (*userid* * * *)
- The specification of L (*userid password*) implies L (*userid password* * *)
- The specification of L (*userid password acct-nmbr*) implies L (*userid password acct-nmbr* *)

The LIST command processor generates the necessary asterisks and also displays that information.

Example 1

Operation: Display the contents of the UADS.

```
list (*)
```

Example 2

Operation: Display all of a particular entry in the UADS.

```
list (wrrid)
```

Example 3

Operation: Display all of the account numbers and procedures under a specific password for a particular entry.

```
l (wrrid roolf *)
```

Example 4

Operation: Display all references to a specific procedure for all entries.

```
l (* * * proc01)
```

ACCOUNT—LISTIDS subcommand

Use the LISTIDS subcommand to display a list of the user IDs in the UADS.

The syntax of the LISTIDS subcommand of ACCOUNT is:

```
LISTI[DS]
```

Example 1

Operation: Display all user IDs in the UADS.

```
listids
```

ACCOUNT—SYNC subcommand

Use the SYNC subcommand to initialize the broadcast data set data set and synchronize it with either the UADS, the TSO/E segment of the RACF data base, or both.

TSO/E copies the user IDs from the UADS and/or the TSO/E segment of the RACF data base into the broadcast data set.

SYNC also formats the NOTICES section of the broadcast data set to reserve room for the maximum number of messages. (Use the IKJBCAST macro to specify the maximum number of messages.)

If you use SYNC when the broadcast data set exists, TSO/E deletes all MAIL from the data set.

In addition, if you use SYNC after you change the message limit for the NOTICES section and the broadcast data set exists (is initialized), the data set is cleared (all MAIL and NOTICES are deleted).

The syntax of the SYNC subcommand is:

```
      ┌ BOTH ┐
      │ ‾‾‾‾ │
SYNC  { RACF }
      │      │
      └ UADS ┘
```

- SYNC is an authorized subcommand.
- To synchronize the broadcast data set with the RACF data base, RACF must be installed and active.

BOTH
: Synchronize the broadcast data set with both the TSO/E segment of the RACF data base and SYS1.UADS, provided that the SYS1.UADS data set was previously allocated to ddname SYSUADS. If it was not previously allocated, the broadcast data set is synchronized with the TSO/E segment of the RACF data base only.

RACF
Synchronize the broadcast data set only with the TSO/E segment of the RACF data base

UADS
Synchronize the broadcast data set only with SYS1.UADS

CONSOLE command

Use the CONSOLE command (along with its subcommands) to perform MVS operator activities from your TSO/E session. (The CONSOLE command can be issued in the background.) The CONSOLE command and its related services require CONSOLE command authority.

The CONSOLE command establishes an extended MCS console session. When the session is active, you can issue MVS system and subsystem commands and obtain command responses and other system messages.

Responses to commands sent through the network to another system might be affected as follows:

- The responses might not be returned as solicited even if a CART was specified and preserved; UNSOLDISPLAY(YES) might be required to receive the responses.
- If the receiving system does not preserve the extended console identifier, ROUTCODE(ALL) and UNSOLDISPLAY(YES) might be required to receive the responses.

For information about ROUTCODE, see the CONSOLxx parmlib member description in *z/OS MVS Initialization and Tuning Reference*. For information about UNSOLDISPLAY, see “CONSPROF command” on page 37.

Throughout this book, the term *console session* means an extended MCS console session established with the CONSOLE command.

Note: Extended MCS console support is provided in line-mode only.

The CONSOLE command is similar to the TSO/E OPERATOR command in that both enable you to perform MVS operator functions. With the OPERATOR command, however, you are limited to performing functions provided only by the OPERATOR subcommands. With the CONSOLE command, the functions you can perform are dependent on the MVS command authority assigned to your console. The command authority defines the types of MVS commands you are authorized to use and is assigned through RACF, by default, or with the MVS VARY command. For more information about MVS command authority, see *z/OS MVS Planning: Operations*.

Preparing for a console session

You can tailor the processing of the CONSOLE command using a console profile. The console profile controls the displaying of messages issued during your console session. You can use the CONSPROF command to change your console profile. For more information about console profiles and the CONSPROF command, see “CONSPROF command” on page 37.

Message retrieval services

If your console profile specifies that messages are not to be displayed at your terminal, applications can retrieve the messages using the GETMSG service. GETMSG is provided as both a programming service and a REXX function. It allows applications to obtain a message from the queue of messages that have been routed to the user's console. For more information about the programming service, see *z/OS TSO/E Programming Services*. For more information about the REXX function, see *z/OS TSO/E REXX Reference*.

Running a console session

Use the CONSOLE command to first activate a console session. When the session is active, you can use the CONSOLE command in either:
- Command mode
- Conversational mode

Using command mode

Command mode is defined as entering the CONSOLE command with one or more keywords.

The console session remains active until the DEACTIVATE keyword is specified on either the CONSOLE command or the END subcommand. The DEACTIVATE keyword terminates the console session.

Using conversational mode

Conversational mode is defined as entering CONSOLE subcommands or system commands in response to the CONSOLE prompt. To enter conversational mode, issue the CONSOLE command without any of the following keywords:
- SYSCMD
- ACTIVATE
- DEACTIVATE

The CONSOLE command then prompts you to enter a subcommand. (TSO/E changes the READY prompt to CONSOLE.)

To exit conversational mode, issue the END subcommand without any operands. Your console session remains active, and TSO/E displays the READY prompt.

Associating commands and their responses

You can write applications to perform MVS operator tasks in the TSO/E environment. The CONSOLE command and the GETMSG service support the use of a command and response token (CART). The CART allows applications to associate commands with their responses. If you set a CART when issuing system commands, your application can then specify the CART on the GETMSG invocation to retrieve responses to specific command invocations.

If multiple applications are to use the CONSOLE command's services in a single TSO/E user's address space, each application must specify a CART to ensure that it retrieves only messages destined for that application. Specifically, the following guidelines should be followed when using the CONSOLE command:

- You should issue all MVS system and subsystem commands with a CART.
- Use the first 4 bytes of the CART as an application identifier. Installations should establish standards so that each program uses an identifier that identifies the program. Whenever the program uses a CART, the CART should begin with the 4-byte identifier.

- You should not display solicited messages at the terminal. Each application should use GETMSG to explicitly retrieve solicited messages intended for that application.

For information about using the CART with the GETMSG service, see *z/OS TSO/E Programming Services* and *z/OS TSO/E REXX Reference*.

Syntax

The syntax of the CONSOLE command is:

```
          ┌                                                                   ┐
          │ ┌ ACTIVATE       ┐  [,NAME(consname)]  [,CART(cartval)]           │
          │ ┤                ├                                                │
          │ └ SYSCMD(syscmd) ┘                                                │
CONSOLE   │ NAME(consname)  [,CART(cartval)]                                  │
          │ CART(cartval)  [,NAME(consname)]                                  │
          │ DEACTIVATE                                                        │
          └                                                                   ┘
```

- If RACF is installed and active, you require either authority to the RACF CONSOLE resource of the RACF TSOAUTH resource class or authority through one of the following installation exit routines to use the CONSOLE command:
  - IKJEFLD or IKJEFLD1
  - IKJCNXAC and IKJCNXCI

  See *z/OS TSO/E Customization* for more information about these exits.
- If RACF is not installed or active, you require authority either through an equivalent security product or through one of the installation exit routines listed above to use the CONSOLE command. In addition, if the CONSOLE command is run in batch, a console name must be specified on the CONSOLE command invocation or via the Console activation exit (IKJCNXAC).

ACTIVATE
Activate a console session with MVS console services. If you do not specify this keyword and a console session was not previously established, activation processing occurs before other CONSOLE command functions are processed.

CART
Set the command and response token (CART). The CART is used to associate commands and their responses. By setting the CART, applications can retrieve responses to specific command invocations using the GETMSG service. You can change the CART for different groups of commands or you can make the CART unique for each system command invocation to distinguish the responses.

If the CONSOLE command is used by multiple applications in a single TSO/E user's address space, you must use the CART to ensure that each application retrieves only messages destined for that application. For specific guidelines, see "Associating commands and their responses" on page 31.

If you do not specify this keyword, the CART is initially set to X'0000000000000000'.

cartval
The command and response token

value:
- 1-8 alphanumeric characters
- 1-16 hexadecimal digits

Note:
1. The CART value is padded to the right with blanks.
2. If you specify a value that is too long, only the first eight bytes (alphanumeric characters or hexadecimal digits) are kept. Any remaining bytes are truncated. No error message is issued.

DEACTIVATE
Terminate the console session.

NAME
Activate the console named *consname*. If you do not specify this keyword, the CONSOLE command uses the user's TSO/E user ID as the console name.

Note: It is recommended for administration purposes that you use your TSO/E user ID as the console name. However, a 1-character user ID cannot be used as a console name because console names must be 2-8 characters in length.

consname
The name of a specific console defined by your installation

value: 2-8 alphanumeric characters, the first of which must be alphabetic or one of the special characters #, $, or @

SYSCMD
Pass a system command to MVS console services for processing

syscmd
The system or subsystem command

value: Any MVS system or subsystem command that you have authorization to issue

CONSOLE command return codes

Table 2. CONSOLE command return codes

0	Request processed successfully.
4	The session is already active and the request is activation.
8	User hit attention.
12	An exit invocation failed.
16	Recovery could not be established.
20	There is no active CONSOLE session and the request is not activation.
24	Request denied. Deactivation in progress.
28	An abend occurred.
32	PUTGET failed.
36	User does not have CONSOLE authority.
40	Activation failed.
44	An exit requested termination.
48	Deactivation failed.
52	Initialization failed.

Table 2. CONSOLE command return codes (continued)

56	NAME specified when already active.
60	Parsing failed.
64	The system command was too long.
68	No CNCCB exists.
72	Process terminated. The level of MVS is not SP 4.1.0 or above.

Example 1

Operation: Activate a console session with MVS console services.

```
CONSOLE ACTIVATE
```

Example 2

Operation: Pass a system command to MVS console services to display the date and time.

```
CONSOLE SYSCMD(D T)
```

Example 3

Operation: Set the command and response token (CART) to associate commands and their responses.

```
CONSOLE CART(X'0000000000000001')
```

Example 4

Operation: Activate a console named NORMAN.

```
CONSOLE NAME(norman)
```

Example 5

Operation: End a console session.

```
CONSOLE DEACTIVATE
```

Subcommands of CONSOLE

To use the subcommands of the CONSOLE command, you must first enter conversational mode. The CONSOLE command enters conversational mode when you do not specify the SYSCMD, ACTIVATE, or DEACTIVATE keywords.

CONSOLE—CART subcommand

Use the CART subcommand to set the command and response token (CART). The CART is used to associate commands and their responses. Applications can use the CART with the GETMSG service to retrieve responses to specific command invocations. See *z/OS TSO/E Programming Services* for information about using the GETMSG programming service and *z/OS TSO/E REXX User's Guide* for information about using the TSO/E REXX external GETMSG function in a REXX exec.

You can change the CART for different groups of commands or you can make the CART unique for each system command invocation to distinguish the responses.

If the CONSOLE command is used by multiple applications in a single TSO/E user's address space, the CART must be used to ensure that each application retrieves only messages destined for that application. For specific guidelines, see "Associating commands and their responses" on page 31.

The syntax of the CART subcommand is:

```
CART cartval
```

cartval
The command and response token

value:
- 1-8 alphanumeric characters
- 1-16 hexadecimal digits

Note:
1. The CART value is padded to the right with blanks.
2. If you specify a value that is too long, only the first eight bytes (alphanumeric characters or hexadecimal digits) are kept. Any remaining bytes are truncated. No error message is issued.

Example 1

Operation: Set the CART to PRT1 and then start printer 1. Reset the CART to PRT2 and then start printer 2.

```
CART PRT1
$S PRT1
CART PRT2
$S PRT2
```

A CART of PRT1 will be returned with all message responses to the $S PRT1 command. A CART of PRT2 will be returned with all message responses to the $S PRT2 command.

CONSOLE—END subcommand

Use the END subcommand to leave conversational mode, and optionally end the current console session. If you specify the END subcommand with no operands, the CONSOLE command leaves conversational mode and the console session remains active. The prompt changes to READY. To end the current console session, issue the END subcommand with the DEACTIVATE (or DEACT) operand.

The syntax of the END subcommand is:

```
     ┌              ┐
     │  DEACTIVATE  │
END  │              │
     │  DEACT       │
     └              ┘
```

DEACTIVATE or DEACT
Terminate the current console session.

Example 1

Operation: Exit CONSOLE conversational mode and remain in the current console session.

```
END
```

Example 2

Operation: Terminate the current console session.

```
END DEACTIVATE
```

CONSOLE—HELP subcommand

Use the HELP subcommand to display a list of valid CONSOLE subcommands or help information for a specific subcommand. If you enter HELP with no subcommand, you see a list of possible subcommands. If you specify a particular subcommand, you see help information for that subcommand. TSO/E provides help information for the following subcommands:
- TSO
- CART
- END

Help information for specific MVS system commands is not provided.

The syntax of the HELP subcommand is:

```
HELP [subcommand]
```

subcommand
The name of a specific subcommand for which you need help information

value: TSO, CART, END

Example 1

Operation: Display help information for the CART subcommand.

```
HELP CART
```

Example 2

Operation: Display a list of valid CONSOLE subcommands.

```
HELP
```

CONSOLE—TSO subcommand

Use the TSO subcommand to issue a TSO/E command from conversational mode. Upon completion of the TSO/E command, you remain in conversational mode.

The syntax of the TSO subcommand is:

```
TSO tso-command
```

tso_command
The name of a specific TSO/E command to be processed

value: any valid TSO/E command

Example 1

Operation: Display your TSO/E profile.

```
TSO PROFILE
```

CONSOLE—system_command subcommand

Use the *system_command* subcommand to issue an MVS system or subsystem command. The command is passed to MVS console services for processing.

The syntax of the *system_command* subcommand is:

```
system-command
```

system_command
The name of the system or subsystem command

value: any MVS system or subsystem command that you have authorization to use

Example 1

Operation: Display the active jobs in the system.

```
D A
```

Example 2

Operation: Display the direct access storage devices (DASDs) that are on-line.

```
D U,DASD,ONLINE
```

CONSPROF command

Use the CONSPROF command to establish, change, or display your console profile. You must have CONSOLE command authority to use CONSPROF.

The information in your console profile is used to control message processing during a console session. You can:

- Specify whether *solicited messages* that are routed to your console are to be displayed at the terminal. A solicited message is a direct response to an MVS system or subsystem command.
- Specify whether *unsolicited messages* that are routed to your console are to be displayed at the terminal. An unsolicited message is any system message that is not a direct response to an MVS system or subsystem command (for example, a message sent to you by another user).
- Assign a value for the maximum number of solicited or unsolicited messages that are to be held for later retrieval with GETMSG.

If you want to receive messages in a language other than U.S. English (specified either by your installation or by using the PROFILE command), you must specify on the CONSPROF command that messages are to be displayed at the terminal. For information about specifying languages with the PROFILE command, see *z/OS TSO/E Command Reference*.

Your installation may have set up a default console profile for you using the logon exit IKJEFLD1. If this has not been done, or if the settings in the profile are not appropriate, you can use the CONSPROF command to change the profile for your console sessions.

If you activate a console session and a profile has not been established (either by your installation or by using the CONSPROF command) both solicited and unsolicited messages that are routed to your console are displayed at the terminal.

To change your console profile, issue the CONSPROF command with the appropriate keywords. Only the keywords specified are updated. If you have RACF installed and you have a TSO segment defined in the RACF data base, the settings defined on the CONSPROF command are maintained from session to session.

To display the current profile settings, issue the CONSPROF command with no operands.

You can use the CONSPROF command during a console session if values need to be changed. If you are in CONSOLE conversational mode, you can use the TSO subcommand of CONSOLE to issue the CONSPROF command.

If you specify that solicited and/or unsolicited messages are not to be displayed at the terminal, applications can use the GETMSG service to retrieve those messages. GETMSG is provided as both a programming service and a REXX function. For more information about using GETMSG, see *z/OS TSO/E Programming Services* or *z/OS TSO/E REXX Reference*.

The syntax of the CONSPROF command is:

```
           ┌                    ┐
           │             {YES}  │
           │ SOLDISPLAY ( {   } )│
           │             {NO }  │
           └                    ┘
           ┌                      ┐
           │               {YES}  │
           │ UNSOLDISPLAY ( {   } )│
           │               {NO }  │
CONSPROF   └                      ┘
           [UNSOLNUM(nnnnn)]

           [SOLNUM(nnnnn)]
```

- CONSPROF is an authorized command.
- You require CONSOLE command authority to use CONSPROF.

SOLDISPLAY(YES | NO)
specifies whether solicited messages that are routed to your console are to be displayed at the terminal.

YES
Solicited messages are displayed at the terminal. This is the default.

NO Solicited messages are not displayed at the terminal. If NO is specified, solicited messages are stored in a message table where you can retrieve them using GETMSG.

SOLNUM(*nnnnn*)
The maximum number of solicited messages that are to be held in a message table. When the limit is approached, installation exits (IKJCNX50 or IKJCNX64) may be invoked to resolve the situation. For a description of what these exits can do, see *z/OS TSO/E Customization*.

If you do not specify this keyword, the system uses either the value that your installation specified in logon exit IKJEFLD1 or the initial value specified in the IKJTSOxx member of SYS1.PARMLIB.

nnnnn is the maximum number of solicited messages; it is an integer in the range of 0 to the maximum value set by your installation in the IKJTSOxx member of SYS1.PARMLIB.

UNSOLDISPLAY(YES | NO)
specifies whether unsolicited messages that are routed to your console are to be displayed at the terminal.

YES
Unsolicited messages are displayed at the terminal. This is the default.

NO Unsolicited messages are not displayed at the terminal. If NO is specified, unsolicited messages are stored in a message table where you can retrieve them using GETMSG.

UNSOLNUM(*nnnnn*)
The maximum number of unsolicited messages that are to be held in a message table. When the limit is approached, installation exits (IKJCNX50 or IKJCNX64) may be invoked to resolve the situation. For a description of what these exits can do, see *z/OS TSO/E Customization*.

If you do not specify this keyword, the system uses either the value that your installation specified in logon exit IKJEFLD1 or the initial value specified in the IKJTSOxx member of SYS1.PARMLIB.

nnnnn is the maximum number of unsolicited messages; it is an integer in the range of 0 to the maximum value set by your installation in the IKJTSOxx member of SYS1.PARMLIB.

CONSPROF command return codes

Table 3. CONSPROF command return codes

0	Processing successful.
4	Processing terminated. A command is invoked without APF authorization.
8	Processing terminated. A command is invoked without console authority.
12	Processing terminated. An error occurred during command buffer parsing.
16	Processing terminated. Recovery could not be established.
20	Processing terminated. Abend occurred during CONSPROF processing.
24	Processing terminated. An installation exit requested CONSPROF termination.
28	Processing terminated. An installation exit had abended.
32	Processing terminated. No CNCCB exists.
72	Processing terminated. The level of MVS is not MVS/ESA SP V4 or above.

Example 1

Operation: Display responses to MVS system and subsystem commands (solicited messages) that are routed to your console.

```
CONSPROF SOLDISPLAY(YES)
```

Example 2

Operation: Do not display unsolicited messages that are routed to your console.

```
CONSPROF UNSOLDISPLAY(NO)
```

Example 3

Operation: Assign 500 as the maximum number of unsolicited messages that are to be held in the unsolicited message table.

```
CONSPROF UNSOLNUM(500)
```

OPERATOR command

Use the OPERATOR command (along with its subcommands) to regulate and maintain TSO/E from a terminal. To authorize the use of OPERATOR by other personnel, specify OPER for a user ID when you create or change an entry in SYS1.UADS; or when you add or alter a user profile in the RACF data base. See *z/OS TSO/E Customization* for additional information about the RACF data base.

The OPERATOR command is not supported under z/OSMF ISPF.

TSO/E fully supports the OPERATOR command only for terminals that have the transmit-interruption capability; that is, it supports this command only for those terminals for which the BREAK parameter of the TERMINAL command is valid.

The syntax of the OPERATOR command is:

```
OPER[ATOR]
```

OPERATOR command return codes

Table 4. OPERATOR command return codes

0	Processing successful.
12	Processing unsuccessful. An error message has been issued.
Other	Return code is from an internal service. Look for any error messages.

OPERATOR—CANCEL subcommand

Use the CANCEL subcommand to terminate the current session of a terminal user. When you use the CANCEL subcommand to terminate the session, the system displays accounting information to the user.

The syntax of the CANCEL subcommand of OPERATOR is:

```
                 ┌                   ┐
                 │      userid       │
C[ ANCEL]  U =  <                     >[ , DUMP]
                 │                   │
                 │*LOGON*, A =asid   │
                 └                   ┘
```

- If a user is currently logged on, specify
  - c u=*userid*

  to terminate that user.
- If a user is attempting to logon and the logon has not completed or cannot complete, the system rejects the command:
  - c u=*userid*

  In this case, issue
  - d ts a

In the display, note the ‘userids’ shown as *LOGON* and their corresponding ASIDs. Then issue
– c u=*logon*,a=*asid*

to terminate a particular user.

- If a logon completes before the CANCEL command takes effect, the system rejects the command:
  – c u=*logon*,a=*asid*

  In this case, reissue
  – c u=*userid*

  to terminate the user.

userid
The user ID of a logged on terminal user whose session you are terminating

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

LOGON
The “userid” of a terminal user attempting to logon

asid
An address space identifier

value: 1-4 hexadecimal digits

DUMP
Take an abnormal-end-of-job storage dump. (The system prints the dump on the system output device.)

Example 1

Operation: Cancel a terminal user's session with a dump.

```
c u=slcid,dump
```

Example 2

Operation: Cancel a terminal user attempting to logon.

```
c u=*logon*,a=002F
```

OPERATOR—DISPLAY subcommand

Use the DISPLAY subcommand to display:
- The time of day and the date
- Summary or detailed information about SLIP traps
- Summary or detailed information about users, jobs, and address spaces
- Summary or detailed information about outstanding requests requiring operator action
- The status of the message processing facility (MPF)
- Information about the status of the global resource serialization complex
- Information about the status of devices
- Summary or detailed information about page and/or swap data sets
- Summary or detailed information about the system-managed storage configuration

The syntax of the DISPLAY subcommand of OPERATOR is:

```
              T
              SLIP[ =xxxx]

              TS        , L[ IST]
              J[ OBS]   , A[ LL]
              A         , name

                     , I
                     , CE         , T
                     , E          , S    [ , KEY =keyname] [ , MSG =msgid]  [ , SYS =sysname]
                     , R          , J
                     , L[ IST]    , M
                R    , ALL

                     , KEY[ , SYS =sysname]

D[ ISPLAY]
                     , U

                                                                                   CONVERSION
                   , M[ SG]                                                        CON
                MPF                                                                C
                   , C[ OLOR]
                                                                                   EXCLUSION
                   , SYSTEM                                                        EXCL
                GRS  , LINK             [ , C[ ONTENTION] [ , HEX] ]     ,RNL =    E            [ , HEX]
                   , A[ LL] [ , HEX]
                                                                                   INCLUSION
                                                                                   INCL
                                                                                   I

                                                                                   A[ LL]
```

```
                           qname                   rname
                           'qname'   [*]           'rname'   [*]
            , RES =(       X'qname'         ,      X'rname'          ) [ , HEX]

                              *                       *

      U   , devicetype   , ONLINE    , dev-num    , nmbr
                         , OFFLINE
          , ALL          , ALLOC     ,  000       , {16 | 8}¹

            , PLPA
            , COMMON
            , DUPLEX
            , LOCAL
            , ALL
D[ ISPLAY]
      ASM                =dsname
            , PAGE
                         =ALL

                         =dsname
            , SWAP
                         =ALL

            , A[ CTIVE]

            , STORGRP        (stor-grp) [ , LISTVOL]
            , SG                       (ALL)
      SMS
            , VOL[ UME] (vol-id)
            , C[ ACHE]
            , T[ RACE]
```

- The specification of a d ts, d j, or d a subcommand displays exactly the same information.
- The first two parameters on a d r subcommand are positional; the remainder are keyword parameters. If you do not specify a positional parameter, indicate its absence by a comma. The following specifications illustrate the rule:
  - d r,,t,msg=iea804e
  - d r,,,msg=iea804e
- All parameters on a d u subcommand are positional. Therefore, if you do not specify a parameter, indicate its absence by a comma. The following specifications illustrate the rule:
  - d u,,alloc
  - d u,tp,,,*nmbr*
  - d u,,,,*nmbr*

If you do not specify either *nmbr* or ALLOC, *nmbr* defaults to 16.

If you specify ALLOC but not *nmbr*, *nmbr* defaults to 8.

- To use the DISPLAY SMS subcommand, the Storage Management Subsystem (SMS) must be installed and active.

T Display the local time of day and the date; and the Greenwich Mean Time (GMT) of day and the date.

SLIP
Display summary information about all the SLIP traps in the system. (The information consists of the trap ids and the state of the traps - enabled or disabled.)

xxxx
Display detailed information about the SLIP trap or traps identified by *xxxx*. (See the SLIP subcommand for details about *xxxx*.)

value:
1. 1-4 alphanumeric characters (a specific trap)
2. 1-3 alphanumeric characters and 1-3 occurrences of an asterisk (*) (a set of traps)
3. **** (all traps)

TS
Display the number of
- active batch jobs
- started tasks (MOUNT commands in execution are treated as started tasks)
- TSO/E users currently logged on
- active system address spaces (e.g., master, global resource serialization, auxiliary, and so on)
- active initiators

If TSO/VTAM is running, display
- the number of users logged on
- the maximum number allowed to be logged on

LIST | L
Include in the display a list of TSO/E user IDs currently logged on and the status of each address space.

ALL | A
Include in the display a list of TSO/E user IDs currently logged on, and, for each address space, also include:
- status
- ASID
- program event recording (PER) active indicator
- number of step-must-complete requests
- performance group number
- domain number
- CPU affinity
- accumulated CPU time
- elapsed time since logon

name
Include in the display only those specified TSO/E user ID(s) currently logged on, and, for each address space, also include:
- status
- ASID
- program event recording (PER) active indicator
- number of step-must-complete requests
- performance group numbers
- domain number

- CPU affinity
- accumulated CPU time
- elapsed time since logon
- names of the data spaces owned by each address space
- real address of the address space number second table (ASTE) for each data space
- real address of the ASTE for each address space

value: TSO/E user IDs

1. 1-7 alphanumeric characters (If you specify a TSO/E user ID as LIST, L, ALL, or A, enclose the user ID in parentheses.)
2. 1-6 alphanumeric characters followed by an asterisk (The display includes all TSO/E user IDs beginning with the specified alphanumeric character(s).)

JOBS | J

Display the number of
- active batch jobs
- started tasks (the system treats MOUNT commands in execution as started tasks)
- TSO/E users currently logged on
- active system address spaces (e.g., master, global resource serialization, auxiliary, and so on)
- active initiators

If TSO/VTAM is running, display
- the number of users logged on
- the maximum number allowed to be logged on

LIST | L

Include in the display, for each active batch job and started task
- jobname
- stepname
- procedure stepname
- V=R region boundaries
- status of each address space

ALL | A

Include in the display, for each active batch job and started task
- jobname
- stepname
- procedure stepname
- V=R region boundaries
- status of each address space

For each address space, also include:
- status
- ASID
- program event recording (PER) active indicator
- number of step-must-complete requests
- performance group number
- domain number
- CPU affinity
- accumulated CPU time
- elapsed time since initiation

name

For each specified batch job or started task, or a specified system address space, include in the display

- for each specified active batch job and started task
  - jobname
  - stepname
  - procedure stepname
  - V=R region boundaries
- for the specified active system address space
  - jobname
  - stepname
  - procedure stepname
- for each of the specified active address spaces
  - status
  - ASID
  - program event recording (PER) active indicator
  - number of step-must-complete requests
  - performance group number
  - domain number
  - CPU affinity
  - accumulated CPU time
  - elapsed time since initiation
  - names of the data spaces owned by each address space
  - real address of the ASTE for each data space
  - real address of the ASTE for each address space

value: Jobnames/started tasks

1. 1-8 alphanumeric characters (If you specify a jobname/started task as LIST, L, ALL, or A, enclose the name in parentheses.)
2. 1-7 alphanumeric characters followed by an asterisk (The display includes all jobnames/started tasks beginning with the specified alphanumeric character(s).)

Specific name of a system address space, e.g.,
- ***
- GRS
- ALLOCAS

A Display the number of
- active batch jobs
- started tasks (the system treats MOUNT commands in execution as started tasks)
- TSO/E users currently logged on
- active system address spaces (e.g., master, global resource serialization, auxiliary, and so on)
- active initiators

If TSO/VTAM is running, display
- the number of users logged on
- the maximum number allowed to be logged on

LIST | L

Include in the display a list of TSO/E user IDs currently logged on and, for each active batch job and started task
- jobname
- stepname
- procedure stepname
- V=R region boundaries
- status of each address space

`ALL | A`

Include in the display

- a list of TSO/E user IDs currently logged on
- for each active batch job and started task
  - jobname
  - stepname
  - procedure stepname
  - V=R region boundaries
- for each active system address space
  - name (e.g., ***, GRS, and so on)
  - stepname
  - procedure stepname
- for every active address space
  - status
  - ASID
  - program event recording (PER) active indicator
  - number of step-must-complete requests
  - performance group number
  - domain number
  - CPU affinity
  - accumulated CPU time
  - elapsed time since logon or initiation

`name`

For each specified TSO/E user ID, batch job, or started task, or a specified system address space, include in the display

- a list of specified TSO/E user IDs currently logged on
- for each specified active batch job and started task
  - jobname
  - stepname
  - procedure stepname
  - V=R region boundaries
- for the specified active system address space
  - name
  - stepname
  - procedure stepname
- for each of the specified active address spaces
  - status
  - ASID
  - program event recording (PER) active indicator
  - number of step-must-complete requests
  - performance group number
  - domain number
  - CPU affinity
  - accumulated CPU time
  - elapsed time since logon or initiation
  - names of the data spaces owned by each address space
  - real address of the ASTE for each data space
  - real address of the ASTE for each address space

value: TSO/E user IDs

1. 1-7 alphanumeric characters (If you specify a TSO/E user ID as LIST, L, ALL, or A, enclose the user ID in parentheses.)

2. 1-6 alphanumeric characters followed by an asterisk (The display includes all TSO/E user IDs beginning with the specified alphanumeric character(s).)

Jobnames/started tasks

1. 1-8 alphanumeric characters (If you specify a jobname or started task as LIST, L, ALL, or A, enclose the name in parentheses.)
2. 1-7 alphanumeric characters followed by an asterisk (The display includes all jobnames and started tasks beginning with the specified alphanumeric character(s).)

Specific name of a system address space, e.g.,
- **MASTER*
- GRS
- ALLOCAS

R Display

- the message id and message text of all
  - outstanding immediate action messages
  - outstanding eventual action messages
  - messages awaiting replies
- the device numbers of all devices
  - with outstanding mount requests
  - awaiting operator intervention
- and the status (active or not active) of the action message retention facility.

I Only include in the display the message id and the message text of all outstanding immediate action messages (descriptor codes 1 and 2).

CE Only include in the display the message id and the message text of all outstanding critical eventual action messages (descriptor code 11).

E Only include in the display the message id and the message text of all outstanding eventual action messages (descriptor code 3).

R Only include in the display the message id and the message text of all messages awaiting replies.

LIST | L
The same display you receive by issuing DISPLAY R

ALL | A
The same display you receive by issuing DISPLAY R

T Include in the display:
- the time a message was issued
- the job id of the issuer
- the name of the system from which the message was issued

S Include in the display:
- the job id of the issuer of a message
- the name of the system from which the message was issued

J Include in the display the job id of the issuer of a message.

M Do not include in the display:
- the time a message was issued
- the job id of the issuer
- the name of the system from which the message was issued

Use the following three keyword parameters
- KEY
- MSG
- SYS

to limit the scope of the message display that you requested through the specification of the first positional parameter. Each of the keyword parameters interact with one another to allow you to tailor the display to include only that information you wish to view.

KEY
Only include in the display the messages associated with *keyname*

keyname
A retrieval key

value: as specified on a WTO/WTOR macro in the KEY parameter

If you do not specify *keyname*, the display includes a summary of keynames, in alphabetical order, associated with the outstanding action messages/WTORs and the number of occurrences for each keyname.

MSG
Only include in the display the id and the text of the outstanding
- immediate action message(s)
- critical eventual action message(s)
- eventual action message(s)
- message awaiting a reply or messages awaiting replies

identified by *msgid*

msgid
A specific message id (for example, IEE427A) or a set of message ids (for example, IEA1)

value: 1-10 alphanumeric characters

SYS
Only include in the display the id and the text of the outstanding
- immediate action message(s)
- critical eventual action message(s)
- eventual action message(s)
- message awaiting a reply or messages awaiting replies

that were issued from the system *sysname*

sysname
The name of a specific system in a complex

value: as specified by your installation

If you do not specify SYS, the display is limited to the system on which TSO/E is running.

KEY
Display a summary of *keynames* and the number of occurrences of each *keyname* for outstanding action messages/ WTORs

SYS
Limit the scope of the display to the system *sysname*

sysname
The name of a specific system in a complex

value: as specified by your installation

If you do not specify SYS, the display is limited to the system on which TSO/E is running.

U Only include in the display the device numbers of all devices with outstanding mount requests and of all devices awaiting operator intervention.

MPF

Display the status of
- MPF
- message suppression
- the action message retention facility
- the general WTO user exit

Also include in the display
- the name(s) of the WTO user exit(s)
- color, highlighting, and intensity options in effect
- if MPF is active
  - include the two-character identifier (xx) and the contents of the MPFLSTxx member of SYS1.PARMLIB currently in effect; and the status of the general WTO user exit
  - if installation-defined color, highlighting, and intensity options are in effect, include the two-character identifier (xx) and the contents of the MPFLSTxx member of SYS1.PARMLIB that defines the options
  - if default color, highlighting, and intensity options are in effect, include the identifier DF and the default options
- if MPF is inactive, include the reason
  - '...NOT INITIALIZED' }MPF processing not requested
  - '...HARD COPY LOG NOT ESTABLISHED'

    MPF processing requested, but no hard copy log available
  - '...HARD COPY SUSPENDED'

    MPF processing requested, but no hard copy log available
    - for the second and third reasons, include the two-character identifier (xx) and the contents of the MPFLSTxx member of SYS1.PARMLIB currently in effect
    - always include the status of the general WTO user exit

MSG | M

Only display information about the messages that are defined in the current MPFLSTxx member of SYS1.PARMLIB:
- which messages MPF is suppressing
- which action messages the action message retention facility is not retaining
- which user exits receive control for selected messages
- the status of the general WTO user exit

COLOR | C

Only display information about the color, highlighting, and intensity options in effect

GRS

Display both system and CTC information for the current global resource serialization complex

SYSTEM

Display system information (For each system in the complex, the display includes system name, state, and communication status.)

LINK

Display CTC information (For each CTC assigned to global resource serialization on this system, the display includes the device number, status, and target system name.)

ALL | A

Display the contents of all RNLs; and resource contention, system, and CTC information for the current global resource serialization complex

HEX

Include in the display the hexadecimal format of the resource name(s) of those resources involved in the resource contention

Note: The system displays all information in EBCDIC format. However, if you specify HEX, the display includes information in both EBCDIC and hexadecimal formats.

CONTENTION | C

Display resource contention information for the current global resource serialization complex

HEX

Include in the display the hexadecimal format of the resource name(s) of those resources involved in the resource contention

RNL

Display the contents of one or more, or all of the RNLs in the current global resource serialization complex

CONVERSION | CON | C

Display the contents of the RESERVE Conversion RNL

EXCLUSION | EXCL | C

Display the contents of the SYSTEMS Exclusion RNL

INCLUSION | INCL | I

Display the contents of the SYSTEM Inclusion RNL

ALL | A

Display the contents of all three RNLs

HEX

Include in the display the hexadecimal format of the resource names contained in the specified RNL(s)

RES

Display a list of major names or resource information for the specified resource(s). For the system to display a list of major names or resource information, the specified resource must have a requestor. Otherwise, the system displays 'NO REQUESTORS FOR RESOURCE _ _ _ _ '.

In the following discussion concerning resource names, appending an asterisk (*) to *qname*/*rname* indicates a set of resources whose major/minor names begin with the specified characters. For example, the specification of SYSV* for *qname* indicates that set of resources whose major names begin with SYSV.

qname

The major name of a resource or the major names of a set of resources

value:

1. 1-8 alphanumeric characters or 1-7 alphanumeric characters and a period (a specific major name)
2. 1-7 alphanumeric characters appended with an asterisk or 1-6 alphanumeric characters and a period appended with an asterisk (a set of major names). For this form of *qname*, if you do not specify *rname*, the system displays just a list of the specified major names of those resources that have requestors.

'*qname*'
The major name of a resource or the major names of a set of resources. *qname* contains a character or characters, other than alphanumeric and the period, from the character set defined in the English (U.S.) I/O Interface Code for 3277, excluding the single quotation mark. (Refer to *IBM 3270 Information Display System Character Set Reference* for more explicit information.) The single quotation marks enclosing *qname* are required; however, they do not count as part of the length specification for *qname*.

value:

1. 1-8 characters, excluding the single quotation mark (a specific major name)
2. 1-7 characters, excluding the single quotation mark, appended with an asterisk following the closing single quotation mark (a set of major names). For this form of '*qname*', if you do not specify *rname*, the system displays just a list of the specified major names of those resources that have requestors.

X'*qname*'
The major name of a resource or the major names of a set of resources. *qname* contains a single quotation mark or multiple single quotation marks and/or characters not included in the character set defined in the English (U.S.) I/O Interface Code for 3277. In this case, specify *qname* in hexadecimal format. The prefix X and the single quotation marks enclosing *qname* are required; however, they do not count as part of the length specification for *qname*.

value:

1. 2-16 hexadecimal digits (a specific major name)
2. 2-14 hexadecimal digits appended with an asterisk following the closing single quotation mark (a set of major names). For this form of X'*qname*', if you do not specify *rname*, the system displays just a list of the specified major names of those resources that have requestors.

***** A generic major name

If you do not specify *rname*, the system displays just a list of the major names of all resources that have requestors.

When you specify *rname* in conjunction with *qname*, the display includes resource information for each unique combination of major-minor name.

rname
The minor name of a resource or the minor names of a set of resources

value:

1. 1-52 alphanumeric characters and periods (a specific minor name)
2. 1-51 alphanumeric characters and periods appended with an asterisk (a set of minor names)

'*rname*'
The minor name of a resource or the minor names of a set of resources. *rname* contains a character or characters, other than alphanumeric and the period, from the character set defined in the English (U.S.) I/O Interface Code for 3277, excluding the single quotation mark. The single quotation marks enclosing *rname* are required; however, they do not count as part of the length specification for *rname*.

value:

1. 1-52 characters, excluding the single quotation mark (a specific minor name)
2. 1-51 characters, excluding the single quotation mark, appended with an asterisk following the closing single quotation mark (a set of minor names)

X'*rname*'
The minor name of a resource or the minor names of a set of resources. *rname* contains a single quotation mark or multiple single quotation marks and/or characters not included in the character set defined in the English (U.S.) I/O Interface Code for 3277. In this case, specify *rname* in hexadecimal format. The prefix X and the single quotation marks enclosing *rname* are required; however, they do not count as part of the length specification for *rname*.

value:

1. 2-104 hexadecimal digits (a specific minor name)
2. 2-102 hexadecimal digits appended with an asterisk following the closing single quotation mark (a set of minor names)

***** A generic minor name

HEX
Include in the display the hexadecimal format of the specified resource name(s)

U Display status information about all device types, including non-supported devices.

devicetype
Only display status information about particular device types

value: as indicated in the following list

- CTC - channel-to-channel adapters
- TP - communications equipment
- GRAPHIC - graphic devices

  Note: In a display that includes graphic devices, the system identifies
  1. a 3290 information panel as a 3279
  2. an MCS console configured as a 3270 model X as a 3270
- TAPE - magnetic tape units
- DASD - direct access storage devices
- UR - unit record devices

ALL
The same display you receive by issuing DISPLAY U

ONLINE
Include in the display only on-line devices

OFFLINE
Include in the display only off-line devices

ALLOC
Include in the display the jobname and ASID of each job to which a device is presently allocated

dev_num
Include in the display only devices whose numbers are equal to or greater than *dev_num*.

value: Three hexadecimal digits

Note:

1. If you specify a device number that you did not specify on the MVSCP IODEVICE statement, the resultant display starts with the next higher device number that you did specify.
2. For multi-exposure devices, the value you specify for *dev_num* must be the same as the value you specified in the ADDRESS parameter of the MVSCP IODEVICE statement.
3. The system displays status information for primary paths only.

nmbr
Include in the display only a specific number of devices

value: 1-4 decimal digits

ASM
For all page and swap data sets, display for each data set
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name

PLPA
Only display information about the PLPA page data set
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name
- volume serial number
- device type
- data set size in slots
- number of slots currently in use
- number of slots currently available
- number of permanent I/O errors that have occurred on the data set

COMMON
Only display information about the common page data set
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name
- volume serial number
- device type
- data set size in slots

- number of slots currently in use
- number of slots currently available
- number of permanent I/O errors that have occurred on the data set

`DUPLEX`
Only display information about the duplex page data set
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name
- volume serial number
- device type
- data set size in slots
- number of slots currently in use
- number of slots currently available
- number of permanent I/O errors that have occurred on the data set

`LOCAL`
Only display information about a local page data set
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name

`ALL`
The same display you receive by issuing DISPLAY ASM

`PAGE`
Only display information about page data sets

`ALL`
For all page data sets, display for each data set
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name

`dsname`
For the specific page data set identified by *dsname*, display
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name
- volume serial number
- device type
- data set size in slots
- number of slots currently in use
- number of slots currently available
- number of permanent I/O errors that have occurred on the data set

value: the name of a specific page data set

`SWAP`
Only display information about swap data sets

ALL

For all swap data sets, display for each data set
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name

dsname

For the specific swap data set identified by *dsname*, display
- type of data set
- percent full
- status (that is, bad, full, or OK)
- device number
- data set name
- volume serial number
- device type
- data set size in swap sets
- number of swap sets currently in use
- number of swap sets currently available
- number of permanent I/O errors that have occurred on the data set

value: the name of a specific swap data set

SMS

Display the active system-managed storage configuration
- the name of the last used source control data set (SCDS)
- the name of the active control data set (ACDS)
- the name of the communications data set (COMMDS)
- the value of the DINTERVAL parameter (See the IGDSMSxx member of SYS1.PARMLIB in *z/OS MVS Initialization and Tuning Guide* for information concerning the DINTERVAL parameter.)
- when the Storage Management Subsystem (SMS) verifies a user's authority to allocate a new data set, use a storage class, or use a management class (See IGDSMSxx in *z/OS MVS Initialization and Tuning Guide* for information concerning the REVERIFY parameter.)
- if SMS will retrieve certain Automatic Class Selection (ACS) defaults from RACF (See IGDSMSxx in *z/OS MVS Initialization and Tuning Guide* for information concerning the ACSDEFAULTS parameter.)
- the value of the INTERVAL parameter for each configuration in the complex (See IGDSMSxx in *z/OS MVS Initialization and Tuning Guide* for information concerning the INTERVAL parameter.)
- a list of the systems in the complex along with the current level of their system-managed storage configuration

ACTIVE | A

The same display as you receive by specifying DISPLAY SMS

STORGRP | SG

Only display storage group information

stor_grp

The name of a specific storage group

value: 1. 1-8 alphanumeric characters, beginning with an alphabetic character

2. if the name of the storage group is ALL, specify it as (ALL)

`LISTVOL`
Include in the display
- a list of the volumes in the storage group
- the status of each volume on every system in the complex that has connectivity to the storage group
- the device number of the volume in the storage group on the system where you issued the DISPLAY subcommand

`ALL`
Only display a list of all storage groups in the system-managed storage configuration

`VOLUME | VOL`
Only display the status of a specific volume on all systems. Include in the display the device number of *vol_id* on the system where you issued the DISPLAY subcommand.

`vol_id`
The name of the volume

value: 1-6 characters. Valid characters are:
- alphanumeric
- - (hyphen)

`CACHE | C`
Only display information about each 3990-3 control unit that has at least one system-managed storage volume attached. The display includes:

SSID Four-character identifier of the subsystem

SMSCT
Number of system-managed storage volumes attached to the cache (3990-3 control unit)

READ CONTROL
Percentage of reads and non-retentive writes that used the cache

FAST WRITE CONTROL
Percentage of writes that used the fast write feature

READ HIT RATIO
Percentage of I/O requests that made a hit in the cache

FAST WRITE RATE
Number of fast write waits per minute

`TRACE | T`
Only display the tracing options in effect (See IGDSMSxx in *z/OS MVS Initialization and Tuning Guide* for an explanation of the tracing options.)

Descriptor Code Meanings

- Action messages with descriptor code 1 - an uncorrectable error occurred and the operator must re-IPL the system or restart a major subsystem.
- Action messages with descriptor code 2 - the operator must perform an action immediately; the issuing task waits until the operator performs the requested action.
- Action messages with descriptor code 3 - the operator must perform an action eventually; the issuing task does not wait for the operator to complete the action.

- Action messages with descriptor code 11 - the operator must perform a critical action eventually; the issuing task does not wait for the operator to complete the action.

Example 1

Operation: Display the number of, and a list of, the TSO/E users currently logged on.

```
d ts,list
```

Example 2

Operation: Display the time of day and the date.

```
d t
```

Example 3

Operation: Display detailed information about SLIP trap 502X.

```
d slip=502x
```

Example 4

Operation: Display the id and the text of the outstanding
- immediate action messages
- critical eventual action messages
- eventual action messages
- messages awaiting replies

identified by the MSG= parameter

```
d r,,,msg=iee
```

Example 5

Operation: Display the status of storage group MYAPPLIC

```
d sms,sg(myapplic)
```

Example 6

Operation: Display the status of storage group ALL

```
d sms,sg((all))
```

Example 7

Operation: Display the active system-managed storage configuration

```
d sms
```

OPERATOR—END subcommand

Use the END subcommand to terminate operation of the OPERATOR command.

The syntax of the END subcommand of OPERATOR is:

```
END
```

OPERATOR—HELP subcommand

Use the HELP subcommand to find out how to use the OPERATOR subcommands. When you enter the HELP subcommand, TSO/E responds by displaying explanatory information at your terminal. You may request:

- A list of available subcommands
- An explanation of the function, syntax, and parameters of a specific subcommand

The HELP subcommand actually causes TSO/E to execute a function of the HELP command; therefore, see the discussion of the HELP command in *z/OS TSO/E Command Reference* if you desire more detailed information.

The syntax of the HELP subcommand of OPERATOR is:

```
         ┌               ┌                                              ┐┐
         │               │ ALL                                          ││
         │               │ FUNCTION                                     ││
         │               │ SYNTAX                                       ││
H[ ELP]  │  subcmd-name  │                                              ││
         │               │                  ┌┌ ┌ƀ┐       ┐      ┐       ││
         │               │ OPERANDS[ (parm1 ││ │,│ parmn ├...) ]│       ││
         └               └                                              ┘┘
```

- If you specify HELP with no parameters, TSO/E displays a list of available subcommands of OPERATOR.

subcmd_name
: The subcommand you want clarified

  value: any valid subcommand of OPERATOR

  ALL
  : Display a description of the function, syntax, positional parameter, and keyword parameters of the subcommand

  FUNCTION
  : Display a description of the function of the subcommand

  SYNTAX
  : Display a description of the proper syntax of the subcommand

  OPERANDS
  : Display a description of the positional and keyword parameters of the subcommand

    parm
    : Display a description of only the indicated keyword parameter(s) of the subcommand

      value: any valid keyword parameter of the subcommand

Example 1

Operation: Display a list of available subcommands at your terminal.

```
help
```

Example 2

Operation: Obtain available information about a particular subcommand.

```
h monitor
```

Example 3

Operation: Display a list of the parameters for a particular subcommand at your terminal.

```
h display operands
```

OPERATOR—MONITOR subcommand

Use the MONITOR subcommand to monitor terminal activities and job activities within the system. The system displays informational messages. The content of the messages pertains to the type of information indicated by the parameter you specify with the MONITOR subcommand. The system continues to issue these informational messages until you issue a STOPMN subcommand or until you terminate the OPERATOR command.

The syntax of the MONITOR subcommand of OPERATOR is:

```
            ┌ ┌JOBNAMES┐          ┐
┌MONITOR┐   │ │        ├ [ , T]  │
│       ├  ┤ └SESS    ┘          ├
└MN     ┘   │                     │
            └  STATUS             ┘
```

JOBNAMES
: Display the name of each job both when the job starts and terminates, and display unit record allocation when the job step starts. If a job terminates abnormally, the jobname appears in the diagnostic message; the message “jobname ENDED” does not appear.

SESS
: Display the user ID whenever a user initiates or terminates a terminal session. If a terminal session terminates abnormally, the user ID appears in the diagnostic message. If the operator cancels a terminal session, the message “user LOGGED OFF” does not appear.

  T Display the local time of day in the following format:

  hh.mm.ss

  where *hh* are the hours (00-23), *mm* are the minutes (00-59), and *ss* are the seconds (00-59).

  Note: After the initial specification (by any user) of the T parameter in the MONITOR subcommand, all subsequent users of MONITOR receive the time of day at their terminals, whether or not they specify T.

STATUS
: Display the names and volume serial numbers of data sets with dispositions of KEEP, CATLG, or UNCATLG whenever the data sets are freed.

Example 1

Operation: Have the system notify you whenever a terminal session begins or ends.

```
monitor sess
```

Example 2

Operation: Display at your terminal the name of each job when the job starts and when it terminates. Also have the time displayed with the jobname.

```
mn jobnames,t
```

OPERATOR—SEND subcommand

Use the SEND subcommand to
- send a message to one or more terminal users
- save a message for retrieval by user(s)
- list, delete, or send a particular message from the notices section of the broadcast data set
- list all messages in the notices section of the broadcast data set
- communicate with console operators and other terminals in operator mode or extended MCS console mode

In a sysplex, SEND can be used to send a message to:
- specific user(s) in the same JESPLEX
- all users on a particular system in the same JESPLEX
- all users on all systems in the same JESPLEX
- all users on a subset of systems in the same JESPLEX

The system appends the characters OPER to the messages you send.

If you issue multiple SEND subcommands, the system may not process them in the order in which you issued them.

To provide better installation control and flexibility in the use of the SEND subcommand, you can modify its operation and function by specifying parameters in the IKJTSOxx member of SYS1.PARMLIB and through the use of OPERATOR SEND exit routines.

Using IKJTSOxx, you can set defaults to
- enable/disable the use of the SEND subcommand
- have messages stored in either the broadcast data set or individual user logs

Using the OPERATOR SEND exit routines, you can
- change the defaults you set in IKJTSOxx
- add diagnostic information to a message
- reformat a message
- change the target data set for a message
- change the target user IDs for a message

For detailed information concerning IKJTSOxx, see *z/OS TSO/E Customization* and *z/OS MVS Initialization and Tuning Guide*. For detailed information concerning the OPERATOR SEND exit routines, see *z/OS TSO/E Customization*.

The syntax of the SEND subcommand of OPERATOR is:

```
              LIST            , ROUTE= { *All        } { , NOW   } { , WAIT   }
                                       { systemname  } { , LOGON } { , NOWAIT }
                                       { groupname   }

                              { , ALL                          } { , LOGON } { , WAIT   }
              { 'msg'     }   { , USER =userid[ , userid]...)  } { , NOW   } { , NOWAIT }
SE [ ND]      { msg-nmbr  }                                      { , SAVE  }

                              { , BRDCST               }
                              { , CN =console          }
                              { , OPERATOR =rte-code   }

              msg-nmbr { , DELETE }
                       { , LIST   }
```

- SEND is an authorized subcommand.
- The system does not write messages to the broadcast data set in the same order as they are sent.

LIST
Display a list of all messages stored in the notices section of the broadcast data set. (Each message displayed is preceded by a system-assigned number.)

msg
The message you are sending. Ensure that the message is a one-line message. If you want a quotation mark as part of the message, enter two quotation marks in the original text.

value: variable-length character string, with a maximum length of 115 characters

msg_nmbr
The identification number of a message in the notices section of the broadcast data set. (The system assigns the identification number.)

value: an integer

ROUTE
Send the message to all users logged on the indicated system(s).

***ALL**
All systems that are members of the same JESPLEX.

systemname
Only system *systemname*

groupname
Only the named subset of systems that are members of the same JESPLEX.

systemname and *groupname* are in MVS name token format. For information on using the MVS program IEEGSYS to define them, see *z/OS MVS Planning: Operations*.

If ROUTE= is not specified, the message is sent only to the users on the system on which the SEND subcommand is issued.

ALL

Send the message to all terminal users.

If you specify ALL, or if ALL is defaulted and ROUTE is not specified, the message will be delivered to all logged on users on the system where the SEND subcommand is issued.

USER

Send the message to the indicated terminal user(s)

userid

The user ID of one or more terminal users who are to receive the message. (The maximum number of user IDs allowed is 20.)

value: 1-7 alphanumeric characters, beginning with an alphabetic or special character

LOGON

Send the message immediately to the terminal user(s) logged on and receiving messages; otherwise,

- users logged on, but not receiving messages, receive it upon requesting messages.
- if you specify ALL, the system stores the message in the notices section of the broadcast data set; sends it to every user as the user logs on and requests messages; and retains it in the broadcast data set until you delete it.
- if you specify USER, the system stores the message in the mail section of the broadcast data set or in the user log data set; sends it to each indicated user as the user logs on and requests messages; and deletes it after all indicated users have received it.
- if you specify NOWAIT, the system creates mail for those users whose terminals are busy.

Using LOGON, a message can be saved for retrieval by a user on any system where the broadcast data set or user log data set is properly shared.

NOW

Send the message immediately

- if you specify ALL, the system sends the message to all terminal users currently logged on, and then deletes it
- if you specify ALL and NOWAIT, users whose terminals are busy do not receive the message. The system does not notify you that the user did not receive the message.
- if you specify USER and NOWAIT, users whose terminals are busy do not receive the message. The system notifies you and then deletes the message.
- if you specify USER, the system sends the message to the indicated terminal user(s) currently logged on. If any indicated terminal user(s) are not logged on, the system notifies you; and then deletes the message. (Terminal users not currently logged on never receive the message.)

SAVE

Store the message in the appropriate section of the broadcast data set. The system does not send the message immediately, even to those terminal users currently logged on and receiving messages.

- if you specify ALL, the system stores the message in the notices section of the broadcast data set, and assigns it an identification number. The system displays the identification number at your terminal. The system sends the message to terminal users as they log on and request messages; and retains it in the broadcast data set until you delete it.

- if you specify USER, the system stores the message in the mail section of the broadcast data set or in the user log data set, and sends it to the indicated terminal user(s) as they log on and request messages. After the last indicated user has received the message, the system deletes it. The WAIT/NOWAIT operand is ignored for SAVE.

Using SAVE, a message can be saved for retrieval by a user on any system where the the broadcast data set data set or user log data set is properly shared.

WAIT
Wait until all specified users can receive the message. If a user's terminal is busy, the other specified users do not receive the message until the user's terminal is no longer busy. A user's terminal is busy if a user's output buffers are full.

NOWAIT
Do not wait until all specified users can receive the message. Even if a user's terminal is busy, the other specified logged on users still receive the message. If you specify USER, you are notified of any users that did not receive the message. If you specify LOGON, the message is saved as mail for those users whose terminals were busy or those users who were not logged on.

BRDCST
Queue the message to all active operator consoles and extended MCS consoles

CN Queue the message to a particular operator console or an extended MCS console

console
For an operator console, specify the console name defined by your installation.

For an extended MCS console, use the name of the console (usually your TSO/E user ID). If you specify an incorrect console name, you receive an error message.

value: a console name, 2-8 alphanumeric characters beginning with either an alphabetic character or one of the special characters: #, $, or @.

INTERNAL
Specifies that the message is to be sent to any active console defined with INTIDS=Y.

For additional information about defining consoles with INTIDS=Y, see *z/OS MVS Planning: Operations*.

OPERATOR
Queue the message to the console associated with the routing code

rte_code
The target destination for a message

value: as shown in the following table

Routing Code	Console
1	Master console action
2	Master console information
3	Tape pool
4	Direct access pool
5	Tape library

Routing Code	Console
6	Disk library
7	Unit record pool
8	Teleprocessing control
9	System security
10	System error/maintenance
11	Programmer information
12	Emulators
13-20	Reserved for your use
21-28	Reserved for subsystem use
29-40	Used by JES3 to represent messages issued for job status
41	Reserved for IBM® use
42	General information about JES2 or JES3
43-64	Reserved for JES2/JES3 use
65-96	Messages associated with particular processors
97-128	Messages associated with particular devices

See *MVS System Messages* (Volume 1 to 10) if you require further explanations of these routing codes.

DELETE
Delete the message identified by *msg-nmbr* from the notices section of the broadcast data set

LIST
Display the message identified by *msg-nmbr* from the notices section of the broadcast data set

Example 1

Operation: Send a message to all terminal users currently logged on. In sysplex systems, the message will be sent to all terminal users currently logged on to any machine in the sysplex that is a member of the same JESPLEX.

```
send 'tso/e to shut down at 9:55 PM. EST 9/14/97'
```

Example 2

Operation: Send a message to all terminal users currently logged on system SYSA in the JESPLEX.

```
send 'tso/e to shut down at 9:55 PM. EST 9/14/97', ROUTE=SYSA
```

Example 3

Operation: Send a message to two particular terminal users currently logged on.

```
send 'your acct no. invalid after this session',user=(heus75,jul65)
```

Example 4

Operation: Delete a message.

```
send 8,delete
```

Example 5

Operation: Display all messages at your terminal.

```
send list
```

Example 6

Operation: Send a message to an extended MCS console named JONES.

```
send 'Please start printer 1',CN=JONES
```

OPERATOR—SLIP subcommand

Use the SLIP subcommand to control SLIP (serviceability level indication processing), a diagnostic aid designed to intercept or trap certain system events. You can indicate what kinds of events you want trapped and what the system should do when these events occur.

The kinds of events you can intercept are:

- Program event recording (PER) events
  - Instruction fetch PER interruption
  - Successful branch PER interruption
  - Storage alteration PER interruption
- Error events
  - Paging error
  - Dynamic address translation error
  - Machine check associated software error
  - Address space termination error
  - SVC 13 issued by a task
  - SVC error
  - Program check interruption
  - Restart interruption

When one of these events occurs, you can take one of the following actions:

- request and tailor an SVC dump specifically to your needs
- cause SLIP to write a GTF trace record
- suppress dumps (for error events only)
- ignore the event
- cause the recovery routines of the interrupted process to get control
- cause SLIP to write a system trace table record
- cause SLIP to write a SYS1.LOGREC record

The PER and error events you can trap are general, and you probably do not want to take one of those actions each time such an event occurs. To narrow the scope of SLIP processing, qualify the event by specifying exactly what state the system must be in when the error or PER event happens for the action to occur. SLIP checks each specified condition to see if it corresponds to the system condition at the time of the error or PER interruption. The conditions you specify serve as filters to screen out those events you are not interested in. When conditions you specify are the same as those in the system, a *match* occurs. When conditions you specify are not the same as those in the system, a *no-match* occurs. Only when a match occurs will SLIP take the specified action. Among the conditions you can specify are:

- the type of error the system is processing
- the system mode at the time of the error or PER interruption
- a user or system completion code associated with the error

- the name of a job or job step program that must be in control at the time of the error or PER interruption
- the module name or address range where the error or PER interruption must occur
- the address space that must be in control at the time of the error or PER interruption
- the contents of specific storage locations and/or registers at the time of the error or interruption

If you do not specify a particular condition, then SLIP makes no checks for that condition.

When you define more than one SLIP trap, SLIP first examines the last defined trap. If it does not find a match condition, it proceeds to check the previously defined trap.

Note that the OPER SLIP command:
- Is limited to 126 characters.
- Cannot use any TSO/E I/O services including the stack.

There are three types of SLIP subcommands:
- SLIP SET subcommand defines SLIP traps.
- SLIP MOD subcommand enables or disables previously defined SLIP traps.
- SLIP DEL subcommand deletes previously defined SLIP traps.

For more information about designing an effective SLIP trap, see *z/OS Problem Management*.

PER monitoring

You can control in which address space, or spaces, PER is active through the specifications of ASID, JOBNAME, and MODE=HOME. The following matrix illustrates the effects of different combinations of specifications. (Note: For information concerning cross memory services and the definitions of particular address spaces, for example, HOME or PRIMARY, see *z/OS MVS Programming: Authorized Assembler Services Guide* or *z/OS MVS Programming: Extended Addressability Guide*.)

ASID	JOBNAME	MODE=HOME	Effect
NO	NO	NO	PER active in all address spaces
NO	YES	NO	PER active in any address space in which the specified job is running
NO	NO	YES	PER active in any address space in which a unit of work is dispatched
NO	YES	YES	PER active in any address space in which the specified job is dispatched
YES	NO	NO	PER active only in the specified address space(s)
YES	YES	NO	PER active in any of the specified address spaces in which the specified job is running
YES	NO	YES	PER active in any of the specified address spaces in which a unit of work is dispatched

ASID	JOBNAME	MODE=HOME	Effect
YES	YES	YES	PER active in any of the specified address spaces in which the specified job is dispatched
: • NO - parameter not specified • YES - parameter specified			

Parameter relationships

SLIP SET parameters fall into six functional groups:
- trap-type parameters
- event filter parameters
- action related parameters
- trap control parameters
- dump and trace tailoring parameters
- specialized parameters

The trap-type parameters are IF, SA, and SB. Each defines a specific type of PER interruption trap. Omitting all trap-type parameters also has meaning: defines an error detection (or non-PER) trap.

The following event filter parameters define the scope of the events, or events, the trap is to monitor:
- ADDRESS
- ASID
- ASIDSA
- COMP
- DATA
- DSSA
- ERRTYP
- JOBNAME
- JSPGM
- LPAEP
- LPAMOD
- MODE
- NUCEP
- NUCMOD
- PSWASC
- PVTEP
- PVTMOD
- RANGE
- REASON

The action related parameter is ACTION. The following operands of ACTION specify what action is to be taken when the trap matches:
- IGNORE
- NODUMP
- NOSUP
- NOSVCD
- NOSYSA
- NOSYSM
- NOSYSU
- RECORD
- RECOVERY

- STDUMP
- STRACE
- SVCD
- SYNCSVCD
- TRACE
- TRDUMP

The following four trap control parameters
- DISABLE
- ENABLE
- MATCHLIM
- PRCNTLIM

Control the operation of the trap by indicating the following:
- whether the trap is active
- how many times the trap should match and produce the desired action before SLIP automatically disables it
- what percentage of system processing SLIP can use when monitoring a PER trap.

The dump tailoring parameters are
- ASIDLST
- DSPNAME
- LIST
- SDATA
- SUMLIST

The trace tailoring parameter is TRDATA.

The parameters enable you to tailor the contents of a dump or a trace record.

The specialized parameters are
- DEBUG
- END
- ID
- RBLEVEL
  - Use DEBUG to diagnose a SLIP trap that is apparently not working according to your specifications; it indicates that you want SLIP to record some trap information each time it checks the trap rather than each time the trap matches.
  - END marks the end of a SLIP SET command.
  - ID assigns an identifier to a trap.
  - RBLEVEL indicates which request block SLIP is to use for error detection traps.

Indirect addressing used with SLIP

Indirect addressing used with SLIP is similar to that used with TSO/E TEST except:
- you may use unlimited levels of indirection
- do not use symbols
- do not follow absolute addresses with a period
- use hexadecimal address modifiers

You can use indirect addresses with the following SLIP subcommand parameters:
- DATA

- LIST
- SUMLIST
- TRDATA

The addresses refer to the address space in which the event occurs unless you specify an address space identifier.

For DATA, SUMLIST, and TRDATA, the storage areas referred to must be paged in. If they are paged out, the system pages them in only if the trap is non-PER and the system at the time of error was unlocked, enabled, and without any EUT FRRs; otherwise SLIP ignores them.

For LIST, the storage areas referred to when resolving the indirect address must be paged in (except for the non-PER, unlocked, enabled, and no EUT FRR case); but the system pages in the storage areas to be dumped if they are paged out.

The elements of an indirect address are:

1. a *direct address*: 1-8 hexadecimal digits optionally followed by one or more displacements.
2. a *displacement*: a plus (+) or minus (-) sign followed by 1-4 hexadecimal digits. The maximum displacement allowed is X'7FFF'.
3. a *general purpose register* or an *access register*: xR where x is an integer in the range 0-15.
4. an *indirection indicator* or *pointer*: a percent sign (%) indicating a 24-bit address or a question mark (?) indicating a 31-bit address. A pointer is always 4 bytes long. (SLIP ignores the high-order byte for 24-bit addresses.)

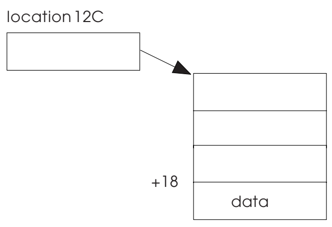
An indirect address is either of the following forms:

```
{direct-address} {%} [%]    [                                  [%] [%]     ]
{              } { } [ ]... [ displacement  [ displacement ]... [ ] [ ]... ]...
{      xR      } {?} [?]    [                                  [?] [?]     ]
```

The following expressions illustrate some indirect addresses.

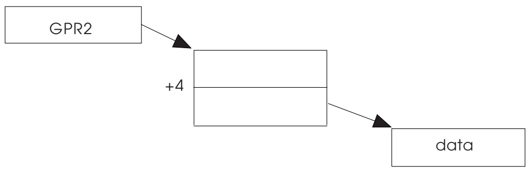
```
12C% +4 +8 +C
```

Graphically:



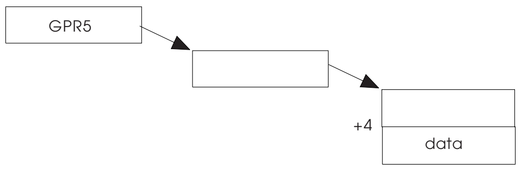
```
2R?+4?
```

Graphically:



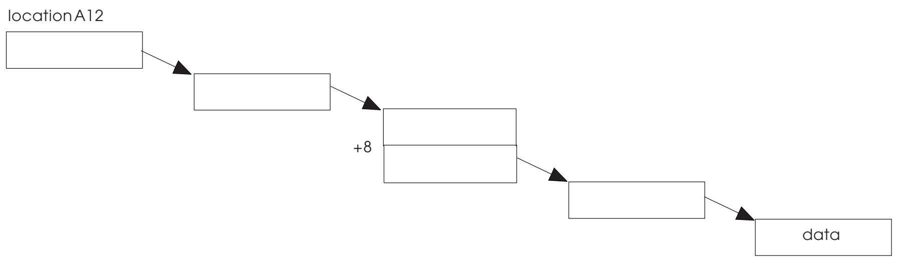
`5R%%+4`

Graphically:

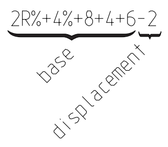


`A12%?+8??`

Graphically:



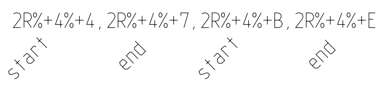
Each complete address is composed of two parts: the base and the displacement. The base is defined as everything except the last displacement. For example, the address `2R%+4%+8+4+6-2` has the following component parts:



If you enter a complete address without a final + or - displacement, SLIP assumes a +0. For example, SLIP treats `2R%+4%` as `2R%+4%+0`.

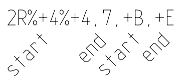
The following discussion applies to the LIST, SUMLIST, and TRDATA parameters when you make multiple *start,end* specifications.

After entering the first complete address (direct or indirect), you can use a form of shorthand for subsequent addresses. The first address establishes the base address. Write subsequent addresses as plus or minus displacements from the base and separate them by commas. For example, you can write the following set of addresses:



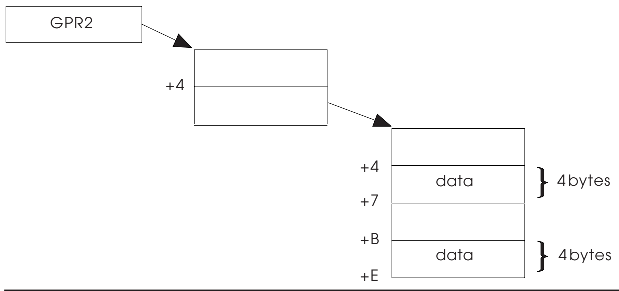
```
2R%+4%+4,2R%+4%+7,2R%+4%+B,2R%+4%+E
```

using the shorthand form as



```
2R%+4%+4,7,+B,+E
```

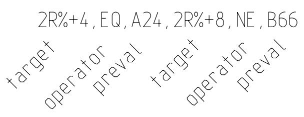
Graphically:



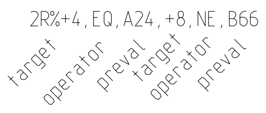
Note: The shorthand form allows you to use control block offsets directly without performing any calculations.

The following discussion applies to the DATA parameter when you make multiple *target* specifications.

After entering the first target address (direct or indirect), you can use a form of shorthand for subsequent target addresses. The first target address establishes the base address. Write subsequent target addresses as plus or minus displacements from the base. For example, you can write the following:
using the shorthand form as



```
2R%+4,EQ,A24,2R%+8,NE,B66
```



```
2R%+4,EQ,A24,+8,NE,B66
```

However, if you specify the first target as the contents of a register, the shorthand form is not valid. For example, the following specification is not valid.

```
2R,EQ,C12,+6,NE,D01
```

After you establish the base address, subsequent target addresses may include both the shorthand form and the contents of a register. For example, the following specification is valid.

```
2R%+4,NE,D11,5R,EQ,15R,+6,GT,C10
```

You may specify an address space identifier as a prefix to a direct or indirect address. If you do not specify the ASID, the address refers to the address space in which the event occurs. After you specify an ASID, the system fetches that address and subsequent addresses within the same keyword from the address space associated with the identifier, until you specify another identifier. An ASID may be:

1. 1-4 hexadecimal digits representing an explicit ASID. (A value of 0 indicates the current address space.)
2. one of the following symbolics

   HASID or H
   home address space

   PASID or P
   primary address space

   SASID or S
   secondary address space

   CURRENT or CU
   current address space

   LLOC locked address space

   SA current alteration space used by a storage alteration trap

   I address space in which the instruction executed

If you do not specify an ASID, SLIP assumes CURRENT.

The following expressions illustrate the use of ASIDs:

```
6.
A.12C%+8%+4
4B.6R%+C%
PASID.12R%+4%+1C
```

In the following expression:

```
6R%+4,+7,SASID.8R%+C,+F,7R%,+1F,+30,+37,H.21C%+10,13
```

- To resolve the `6R%+4,7` addresses, SLIP uses the address space in which the event occurred (current address space).
- To resolve the `8R%+C,+F,7R%,+1F,+30,+37` addresses, SLIP uses the secondary address space at the time the event occurred.
- To resolve the `21C%+10,13` addresses, SLIP uses the home address space at the time the event occurred.

When you use register notation for an address, SLIP normally uses the contents of a GPR to calculate the address. However, if all the following conditions are true, SLIP resolves an address using the address space or data space indicated by an access register:

- the processor is operating in access register (AR) ASC mode
- the indirect address begins with register notation
- CURRENT is specified or implied

SLIP continues to use that space until you explicitly change it by specifying CURRENT.

In the following expression:

```
CU.3R%,4R%,5R%,6R%,CU.7R%,8R%,CU.8000,+4
```

- To resolve the `3R%,4R%` addresses, SLIP uses the space indicated by access register 3.
- To resolve the `5R%,6R%` addresses, SLIP again uses the space indicated by access register 3.
- To resolve the `7R%,8R%` addresses, SLIP now uses the space indicated by access register 7.
- To resolve the `8000,+4` addresses, SLIP uses the primary space. (When you do not use register notation, SLIP uses the primary space to resolve addresses.)

Note: The specification of an address space identifier as a prefix to the shorthand form of an indirect address is not valid. For example, the following is a not valid specification:

```
7R%,+1F,PASID.+30,+37
```

Parameter descriptions

The major positional and keyword parameters are described in alphabetical order. The subparameters are described under the major parameters in alphabetical order.

- Parameters
  - DSPNAME
  - DSSA
  - PSWASC
  - SYNCSVCD
- Address Space Identifiers
  - I
  - SA

ACTION | A

The action that is to occur when a trap matches.

IGNORE

Resume normal processing when the trap matches

NODUMP

Suppress the following dumps
- SVC dumps requested by ESTAE and FRRs
- all SYSABEND, SYSUDUMP, and SYSMDUMP dumps

Note:

1. When you specify ACTION=NODUMP, ensure that the SLIP trap is specific. If the trap is too general, you might suppress dumps needed for other problems. For example, if you specify only a system completion code, SLIP suppresses all dumps for that code. However, if you specify both a completion code and a jobname, other jobs that abend with that completion code produce dumps.
2. If a second error occurs during processing for an event with ACTION=NODUMP specified, SLIP also suppresses any dump you request for the second error. You can determine if a second error occurred by checking both the job output messages and SYS1.LOGREC output. If either one indicates more than one abend, a second error occurred. If you need a dump for the second error, disable the SLIP trap that specifies ACTION=NODUMP and rerun the failing job.

NOSUP

1. Override the action of dump analysis and elimination (DAE) in suppress mode and do *not* suppress any duplicate SVC or SYSMDUMP dumps.
2. Override the specification of dump suppression requested by a user ABDUMP predump exit and do *not* suppress any SYSUDUMP or SYSABEND dumps.

NOSVCD
Suppress only SVC dumps requested by ESTAE and FRRs when the trap matches.

NOSYSA
Suppress only requested SYSABEND dumps when the trap matches.

NOSYSM
Suppress only requested SYSMDUMP dumps when the trap matches.

NOSYSU
Suppress only requested SYSUDUMP dumps when the trap matches.

RECORD
Force recording to SYS1.LOGREC for every recovery routine, regardless of what the recovery routine specifies.

RECOVERY
Force PER traps to initiate recovery processing for the interrupted process after the specified action is taken. (System completion code 06F is generated.)

Note: Use the RECOVERY keyword carefully to avoid unexpected results. Before using RECOVERY, be thoroughly familiar with MVS recovery principles. In particular, ensure that recovery procedures exist at the point where you are forcing recovery processing. Know what the recovery routines will do under the circumstances in which you are forcing recovery processing.

STDUMP
Create SLIP system trace records while the trap is enabled, and schedule an SVC dump when the trap is disabled or you delete it. (System trace must be active when you specify STDUMP.)

The specification of STDUMP always overrides the duplicate dump suppression action requested through DAE.

Note: Although you can use all keywords that are valid on PER traps on a SLIP trap with ACTION=STDUMP, the use of only the following keywords requires less SLIP processing (the use of less system resources):

IF and SB traps
ENABLE/DISABLE, ID, MATCHLIM, RANGE/NUCEP/NUCMOD/LPAEP/LPAMOD, ASIDLST, LIST, SDATA, SUMLIST, and END

SA traps
ENABLE/DISABLE, ID, MATCHLIM, RANGE, ASIDLST, LIST, SDATA, SUMLIST, and END

If you do not specify MATCHLIM, SLIP disables the trap after 50 matches.

However, if you specify keywords other than those indicated and do not specify MATCHLIM, a ‘no limit’ to the number of trap matches exists.

STRACE
Write at least one SLIP system trace record when the trap matches. For SLIP to write the record, system trace must be active.

Note: Although you can use all keywords that are valid on PER traps on a SLIP trap with ACTION=STRACE, the use of only the following keywords requires less SLIP processing (the use of less system resources):

IF and SB traps
ENABLE/DISABLE, ID, MATCHLIM, RANGE/NUCEP/ NUCMOD/LPAEP/LPAMOD, and END

SA traps
ENABLE/DISABLE, ID, MATCHLIM, RANGE, and END

If you do not specify MATCHLIM, SLIP disables the trap after 50 matches.

However, if you specify keywords other than those indicated and do not specify MATCHLIM, a 'no limit' to the number of trap matches exists.

SVCD
Schedule an SVC dump when the trap matches. (If an address space is failing and you did not specify a list of ASIDs to be dumped, SLIP tries to dump the failing address space. If SLIP cannot dump the failing address space, it dumps the current address space.)

The specification of SVCD always overrides the duplicate dump suppression action requested through DAE.

SYNCSVCD
Schedule a synchronous SVC dump when the trap matches. (If an address space is failing and you did not specify a list of ASIDs to be dumped, SLIP tries to dump the failing address space. If SLIP cannot dump the failing address space, it dumps the current address space.)

The specification of SYNCSVCD always overrides the duplicate dump suppression action requested through DAE.

When all the following conditions exist:
1. a PER interrupt occurs
2. the system is enabled and unlocked
3. the system is in task or SRB mode
4. the system is using the normal FRR stack

SLIP stops the unit of work before starting the dump to ensure that the restart occurs after the dump completes. If the system is disabled or locked when the PER interrupt occurs, SLIP schedules a regular SVC dump.

TRACE
Create a SLIP GTF trace record when the trap matches.

SLIP may write the record to external storage or maintain it in virtual storage according to the GTF options you select. (For detailed information on GTF functions, see *z/OS MVS Diagnosis: Tools and Service Aids*.)

If you do not specify TRDATA, SLIP creates a SLIP standard trace record when the trap matches. (For TRACE to be active, GTF with the SLIP option must be active.)

TRDUMP
Create a SLIP GTF trace record each time a trap matches and schedule an SVC dump when you delete or disable the trap.

If you do not specify TRDATA, SLIP creates a SLIP standard trace record when the trap matches. (For TRDUMP to be active, GTF with the SLIP option must be active.)

The specification of TRDUMP always overrides the duplicate dump suppression action requested through DAE.

`TRDATA | TD`
Tailor the type and contents of a SLIP GTF trace record

`STD`
Create a standard SLIP GTF trace record when the trap matches

`REGS`
Collect the contents of the 16 GPRs and the 16 access registers into the SLIP GTF trace record when the trap matches

`asid`
An address space identifier

value:

1. 1-4 hexadecimal digits (Do not specify a value that exceeds the maximum value set by your installation.)
2. as indicated in the following list:

HASID or H	home address space
PASID or P	primary address space
SASID or S	secondary address space
CURRENT or CU	current address space
LLOC	locked address space
SA	current alteration space used by an SA trap
I	address where the instruction executed

`start``,`*`end`*
Collect the contents of the address range, or ranges, from the address space, or spaces, into the SLIP GTF trace record when the trap matches. The address range can be:

1. a virtual address (direct address)
   value: 1-8 hexadecimal digits for *start* and *end*. (Specify an ending address that is greater than or equal to the starting address.)
2. an indirect address
   value: see “Indirect addressing used with SLIP” on page 69. (Specify an ending address that is greater than or equal to the starting address.)

Do not specify an address range larger than 65,535 bytes. If any range exceeds 65,535 bytes, SLIP does not write any data into the trace record but writes a zero-length indicator to indicate the error.

`ASIDLST | AL`
Dump the address space, or spaces, when the trap matches.

n An address space identifier. (The maximum number of identifiers allowed is 15.)

value:

1. 1-4 hexadecimal digits. (Do not specify a value that exceeds the maximum value set by your installation.)
2. as indicated in the following list:

HASID or H	home address space
PASID or P	primary address space
SASID or S	secondary address space
CURRENT or CU	current address space
LLOC	locked address space
SA	current alteration space used by an SA trap
I	address space where the instruction executed

If you specify SA and the alteration space is a data space, SLIP dumps the data space.

DSPNAME

Include the data space, or data spaces, in an SVC dump when the trap matches

asid

An address space identifier (The maximum number of identifiers allowed is 15.)

value:

1. 1-4 hexadecimal digits (Do not specify a value that exceeds the maximum value set by your installation.)
2. as indicated in the following list:

HASID or H	home address space
PASID or P	primary address space
SASID or S	secondary address space
CURRENT or CU	current address space
LLOC	locked address space
SA	current alteration space used by a storage alteration trap
I	address space where the instruction executed

name

The name you used to create the data space

value: 1-8 alphanumeric characters

When the interrupted unit of work holds a lock higher than the RSM lock, the system cannot determine the specific data space(s). Therefore, SLIP does not include any data spaces in the dump.

LIST | LS

Include the address range, or ranges, from the address space, or spaces, in an SVC dump when the trap matches

asid

An address space identifier

value:

1. 1-4 hexadecimal digits (Do not specify a value that exceeds the maximum value set by your installation.)
2. as indicated in the following list:

HASID or H
home address space

PASID or P
primary address space

SASID or S
secondary address space

CURRENT or CU
current address space

LLOC locked address space

SA current alteration space used by a storage alteration trap

I address space where the instruction executed

start,*end*

The starting and ending addresses. The address range can be:

1. a virtual address (direct address)

   value: 1-8 hexadecimal digits for *start* and *end*. (Specify an ending address that is greater than or equal to the starting address.)
2. an indirect address

   value: see "Indirect addressing used with SLIP" on page 69. (Specify an ending address that is greater than or equal to the starting address.)

Two error conditions can arise when using LIST. The first involves the resolution of an indirect address. If, for any reason, SLIP cannot convert an indirect address to a direct address (for example, a page fault occurs while SLIP is retrieving a pointer or registers are unavailable for conversion), SLIP dumps the characters *RC=4* instead of the address range requested to indicate that it could not successfully convert the address pair.

The second condition occurs after SLIP successfully converts a pair of indirect addresses, but the second address (ending address) is less than the first address (starting address) of the pair. SLIP dumps the characters *A1>A2* instead of the address range requested (prevents SDUMP from abending).

`SDATA | SD`

Include system control information in an SVC or summary dump when the trap matches

option

An area of storage or type of dump

value: as indicated in the following list:

ALLNUC	all of the DAT-on and DAT-off nuclei
ALLPSA	prefix storage area for all central processors
CSA	common storage area
GRSQ	global resource serialization queues
LPA	link pack area
LSQA	local system queue area
NOALLPSA or NOALL	not ALLPSA
NOSQA	not SQA
NOSUMDUMP or NOSUM	not SUMDUMP
NUC	only the non-page protected part of the DAT-on nucleus
PSA	prefix storage area of dumping central processor
RGN	entire private area
SQA	system queue area
SUMDUMP or SUM	summary dump function
SWA	scheduler work area
TRT	GTF or supervisor trace data

- For ACTION=SVCD, if you do not specify SDATA, SLIP assumes the following:
  ```
  SDATA=(ALLPSA,CSA,LPA,NUC,RGN,SQA,SUM,TRT)
  ```
- For ACTION=TRDUMP or =STDUMP, if you do not specify SDATA, SLIP assumes the following:
  ```
  SDATA=(TRT,NOALLPSA,NOSQA,NOSUM)
  ```
- If you explicitly specify any SDATA options, SLIP ignores any alterations to those options specified on the CHNGDUMP command. However, if CHNGDUMP is set with the NODUMP option, SLIP does not produce a dump when the trap matches. (For more detailed information relative to SDATA, CHNGDUMP, and SDUMP, see *z/OS Problem Management*.)

`SUMLIST | SL`

Include the address range, or ranges, from the address space, or spaces, in a summary dump when the trap matches. (If you specify SDATA=(NOSUMDUMP), the specification of SUMLIST is not valid.)

`asid`
An address space identifier. (If you do not specify *asid*, SLIP assumes current.)

value:

1. 1-4 hexadecimal digits (Do not specify a value that exceeds the maximum value set by your installation.)
2. as indicated in the following list:

   HASID or H
   home address space

   PASID or P
   primary address space

   SASID or S
   secondary address space

   CURRENT or CU
   current address space

   LLOC locked address space

   SA current alteration space used by an SA trap

   I address space where the instruction executed

`start,end`
An address range. The address range can be:

1. a virtual address (direct address)

   value: 1-8 hexadecimal digits for *start* and *end*. (Specify an ending address that is greater than or equal to the starting address.)
2. an indirect address

   value: see "Indirect addressing used with SLIP" on page 69. (Specify an ending address that is greater than or equal to the starting address.)

Two error conditions may arise when using SUMLIST. The first involves the resolution of an indirect address. If, for any reason, SLIP cannot convert an indirect address to a direct address (for example, a page fault occurs while SLIP is retrieving a pointer or registers are unavailable for conversion), SLIP dumps the characters *RC=4* instead of the address range requested to indicate that it could not successfully convert the address pair.

The second condition occurs after SLIP successfully converts a pair of indirect addresses, but the second address (ending address) is less than the first address (starting address) of the pair. SLIP dumps the characters *A1>A2* instead of the address range requested (prevents SDUMP from abending).

`ADDRESS | AD`
The event must occur at a virtual address, or within a range of virtual addresses, to satisfy the match test

`start`
A virtual address (1-byte range)

value: 1-8 hexadecimal digits

`start,end`
A virtual address range

value: 1-8 hexadecimal digits for *start* and *end* (Specify an ending address that is greater than or equal to the starting address.)

- For more information on choosing the virtual address or address range relative to the environment, see *z/OS Problem Management*.

ASID | AS
The event must occur within an address space, or spaces, to satisfy the match test. For storage alteration PER traps, this keyword refers to the address space, or spaces, from which the system fetches the instructions. See "PER monitoring" on page 67 for additional information.

id An address space identifier. (The maximum number of ids allowed is 16.)

value: 1-4 hexadecimal digits. (Do not specify a value that exceeds the maximum value set by your installation.)

ASIDSA | ASA
The storage being altered must reside within an address space, or spaces, to satisfy the match test.

asid
An address space identifier. (The maximum number of ids allowed is 16.)

value:

1. 1-4 hexadecimal digits (Do not specify a value that exceeds the maximum value set by your installation.)
2. as indicated in the following list:

HASID or H	home address space
PASID or P	primary address space
SASID or S	secondary address space
CURRENT or CU	current address space
LLOC	locked address space
SA	current alteration space used by an SA trap
I	address space where the instruction executed

COMP | C
Associate a system completion code or user completion code with an error. If you specify a set of codes, the occurrence of any one satisfies the match test.

hhh
A system completion code or a set of system completion codes

value:

1. 3 hexadecimal digits (a unique code)
2. 0-2 hexadecimal digits and 1-3 occurrences of X (a set of codes), valid specifications - XxX, xXX, xxX, XXx, xXx, Xxx, XXX (For example, X11 means 011, 111, ..., F11)

Note:

1. If you specify any of the following system completion codes, the match test always fails:

   ```
   11A,12E,15D,15F,200,212,279,25F,282,42A,57D,6FC,700,
   72A,A00,B00,E00,X22
   ```

   Most of the preceding codes occur originally as a program check (0C4) that the system converts to the indicated code. SLIP can detect the original code (0C4), but not the converted code. To specify a program check, use COMP=0C4 or ERRTYP=PROG. To avoid satisfying the match test for all program checks, specify a program name, module name, or other qualifier.
2. The specification of 13E, 33E, or 922 prevents a trap match because those completion codes occur for any active subtasks associated with a task that is abending. The secondary abends occur for the purpose of cleanup only and SLIP does not detect them.
3. For such abend codes as 201, 202, 402, and 702, the SLIP action might not be taken. In certain paths, each of those codes was originally a program check. In other paths, the abend was issued directly. To ensure the SLIP action is taken, set, for example, one SLIP trap specifying COMP=201 and another specifying COMP=0C4 or ERRTYP=PROG.
4. If a recovery routine, using the SETRP macro, changes any abend code, specify the original code in the SLIP command. For example, specify COMP=171 instead of COMP=800.

Udddd

A user completion code or a set of user completion codes

value:

1. 4 decimal digits (a unique code)
2. 0-3 decimal digits and 1-4 occurrences of X (a set of codes), valid specifications -

UdddX	UXXdd	UddXd	UXdXd
UddXX	UXXXd	UXXdX	UdXdX
UdXXX	UdXdd	UXdXX	UXXXX
UXddd	UdXXd	UXddX	

Note: If a user recovery routine, using the SETRP macro, changes any user completion code, specify the original completion code in the COMP keyword parameter.

REASON | RE

Associate a reason code with the error

code

A reason code or a set of reason codes

value:

1. 1-8 hexadecimal digits (a unique code)
2. 1-7 hexadecimal digits and 1-7 occurrences of X (a set of reason codes). SLIP ignores the digit(s) of a reason code specified as an X.

Note:

1. The specification of REASON is valid only if a user coded the REASON parameter on the ABEND, SETRP, or CALLRTM macro instruction.

2. If **code** is less than eight digits, SLIP pads it on the left with zeroes. For example, SLIP stores REASON=4 as 00000004; REASON=XX0X1C as 00XX0X1C.

DATA | DA

Logically compare the contents of a target location to a specified value. The complete logical expression must be satisfied to satisfy the match test. (If a DATA=*target* is paged out, SLIP assumes a no match; issues message IEA413I; and updates a "data unavailable" count in the SCVA. For a PER trap, SLIP notifies you only the first time the data is unavailable. However, the "data unavailable" count is readily available by displaying the trap or can be found in the standard portion of a SLIP trace record.)

asid

An address space identifier. (If you do not specify *asid*, SLIP assumes current.)

value:

1. 1-4 hexadecimal digits. (Do not specify a value that exceeds the maximum value set by your installation.)
2. as indicated in the following list:

HASID or H
home address space

PASID or P
primary address space

SASID or S
secondary address space

CURRENT or CU
current address space

LLOC locked address space

SA current alteration space used by an SA trap

I address space where the instruction executed

target

The address of a storage location or a general purpose register (GPR). *target* can be:

1. a virtual address (direct address)

   value: 1-8 hexadecimal digits
2. a GPR in the form *x*R

   x A register designation

   value: an integer in the range 0-15
3. an indirect address

   value: see "Indirect addressing used with SLIP" on page 69

b Target modifier that indicates the starting bit position for a *binary* comparison

value:

1. an integer in the range 0-7 (storage locations)
2. an integer in the range 0-31 (GPRs)

operator

A logical operator

value: as indicated in the following list:

EQ	equal
NE	not equal
GT	greater than
LT	less than
NG	not greater than (less than or equal to)
NL	not less than (greater than or equal to)

`A - address compare`
Compare the *target* address with the *preval* address. SLIP ignores the high-order bit in the compare. (Do not specify *b* for an address compare.)

`C - contents compare`
Compare the contents of the *target* address with the contents of the *preval* address

n Number of bits or bytes involved in the compare

value:

1. 1-8 bits for a binary compare (the default is 1 bit)
2. 1-4 bytes (the default is 4 bytes)

When SLIP performs an address or contents compare, it uses the leftmost *n* bytes of storage, and the high-order (rightmost) *n* bytes of a register.

`preval`
The data to compare with the *target*

value: - A or C not specified:

1. binary digits - maximum length of 8 bits (*b* specified). (The comparison can be across a byte boundary, but not across a register boundary.)
2. hexadecimal digits - maximum length of 4 bytes, right justified (*b* not specified)

- The length of *preval* determines the length of the compare except for the hexadecimal compare of a register. In that case, SLIP right justifies *preval*, pads it with zeros; and the compare length is the entire register.

value: - A or C specified

1. a virtual address (direct address)

   value: 1-8 hexadecimal digits
2. a GPR in the form *x*R

   x A register designation

   value: an integer in the range 0-15
3. an indirect address

   value: see “Indirect addressing used with SLIP” on page 69

DEBUG
provides information to allow you to determine why a trap you set is not working as you expected. (For DEBUG to be active, GTF with the SLIP option must be active.)

Each time SLIP *tests* the trap, it writes a trace record, containing the standard SLIP trace data plus two bytes of match/no match bit indicators. Each bit corresponds to a possible test SLIP made to determine a match for the trap. If a test is successful, SLIP sets the corresponding indicator to 0. If a test is unsuccessful, SLIP sets the corresponding indicator to 1. After the first unsuccessful test, SLIP makes no further tests and sets all the remaining indicators to 0. For a description of the SLIP DEBUG trace record and the bit indicators, see *z/OS MVS Diagnosis: Tools and Service Aids* or *z/OS MVS Diagnosis: Reference*.

DISABLE | D
Initially disable a defined SLIP trap

ENABLE | EN
Initially enable a defined SLIP trap

DSSA
The storage being altered must reside within a data space, or spaces, to satisfy the match test.

asid
An address space identifier. (The maximum number of ids allowed is 16.)

value:

1. 1-4 hexadecimal digits. (Do not specify a value that exceeds the maximum value set by your installation.)
2. as indicated in the following list:

HASID or H
home address space

PASID or P
primary address space

SASID or S
secondary address space

CURRENT or CU
current address space

LLOC locked address space

SA current alteration space used by a storage alteration trap

I address space where the instruction executed

name
The name you used to create the data space

value: 1-8 alphanumeric characters

Note:

1. If you specify SA.*name*, the storage alteration must occur in the named data space for the trap to match.
2. If you specify SA only and the storage alteration occurs in the address space, the trap does not match.

3. If you specify *asid* only, the trap matches on a storage alteration in any data space owned by the specified *asid*.

END | E
The end of a SLIP SET command. If you do not specify END, the system prompts you for additional keywords.

Note: If you enter a SLIP command from a CLIST, set multi-line SLIP commands in the CLIST using the line continuation character (a blank). End the command with the END positional parameter on the last line.

ERRTYP | ER
An error condition must occur to satisfy the match test. If you specify more than one error condition, the occurrence of any one satisfies the match test.

type
An error condition

value: as indicated in the following list:

ALL	all of the following error conditions
ABEND	task issued SVC13
DAT	dynamic address translation error
MACH	software error caused by machine check
MEMTERM	abnormal address space termination
PGIO	paging I/O error
PROG	program check interruption
REST	restart interruption
SVCERR	SVC error (issuing an SVC while holding a lock, executing disabled, or executing in SRB mode)

ID Assign an identifier to a trap. If you do not specify ID, the system assigns a unique four-character identifier beginning with 0001. The system issues message IEE727I to notify you of the assigned identifier.

xxxx
Trap identifier

value: 1-4 alphanumeric characters

IF Monitor an instruction fetch PER trap

JOBNAME | J
The initiated job, started task, or TSO/E session that must be in control to satisfy the match test. See "PER monitoring" on page 67 for additional information.

job_name
The jobname, started task id, or TSO/E user ID. (The *job_name* is the one specified on the JOB statement; the started task id is the *procname* specified on the START command; the TSO/E user ID is the *userid* specified on the LOGON command.)

value:

1. 1-8 alphanumeric characters, beginning with an alphabetic or special character (jobname and started task id)
2. 1-7 alphanumeric characters, beginning with an alphabetic or special character (TSO/E user ID)

`JSPGM | JS`

The job step program that must be in control to satisfy the match test. If you specify JSPGM for an error trap and any address space abnormally terminates, a no-match condition for the trap occurs.

`js_name`

The job step program name. (This name is the one specified on the EXEC statement in the PGM=*program-name* parameter.)

value: 1-8 alphanumeric characters, beginning with an alphabetic or special character

`LPAEP`

The event must occur within a link pack area load module, relative to the specified entry point, to satisfy the match test.

- If you do not specify *start* and *end*, SLIP monitors a range from the specified entry point to the end of the module.
- If you specify only *start*, SLIP monitors a 1-byte range from the offset *start*.
- If you specify both *start* and *end*, SLIP monitors the range between the offset *start* and the offset *end*.

`mod_ep`

The entry point name

value:

1. 1-8 alphanumeric characters
2. 1-7 alphanumeric characters appended with an asterisk. (SLIP interprets the asterisk as X'C0'.)

`start`

The offset into the module from the entry point (1-byte range)

value: 1-8 hexadecimal digits

*`start`***`,`***`end`*

The starting and ending offsets into the module

value: 1-6 hexadecimal digits for *start* and *end*. (Specify a value for *end* that is greater than or equal to that of *start*)

`LPAMOD | L`

The event must occur within a link pack area load module to satisfy the match test.

- If you do not specify *start* and *end*, SLIP monitors the entire module.
- If you specify only *start*, SLIP monitors a 1-byte range from the offset *start*.
- If you specify both *start* and *end*, SLIP monitors the range between the offset *start* and the offset *end*.

`mod_name`

The module name

value:

1. 1-8 alphanumeric characters
2. 1-7 alphanumeric characters appended with an asterisk. (SLIP interprets the asterisk as X'C0'.)

start
The offset into the module (1-byte range)

value: 1-6 hexadecimal digits

*start***,***end*
The starting and ending offsets into the module

value: 1-6 hexadecimal digits for *start* and *end*. (Specify a value for *end* that is greater than or equal to that of *start*.)

- For more information on choosing the *start* and *end* offsets, see *z/OS Problem Management*.

MATCHLIM | ML
Automatically disable an enabled trap after the specified number of matches.

- If you specify ACTION=SVCD or =SYNCSVCD for a PER trap and do not specify MATCHLIM, SLIP assumes 1.
- If you specify ACTION=STDUMP or =STRACE for a PER trap and do not specify MATCHLIM, SLIP assumes 50.
- For all other traps, if you do not specify MATCHLIM, SLIP assumes no limit; that is, it makes no MATCHLIM test.

When the specified number of trap matches occurs, SLIP disables an enabled trap and issues message IEA411I. If you specified ACTION=TRDUMP, SLIP schedules an SVC dump.

m Number of trap matches

value: integer in the range 1-65535

MODE | M
The system must be in a particular mode, or modes, to satisfy the match test.

cond
A system mode

value: as indicated in the following list:

HOME	executing in the home (dispatched) address space
ALL	all of the following modes
DIS	physically disabled for I/O and external interruptions
GLOC	holding any global lock
GLOCSD	holding a global suspend lock
GLOCSP	holding a global spin lock
LLOC	holding a local lock
LOCK	holding any lock
PKEY	problem program key (key 8 or higher)
PP	problem program
RECV	recovery routine in control (RECV is a not valid specification for all PER traps.)
SKEY	system key (key 0-7)

SRB SRB mode

SUPER
supervisor state

SUPR supervisor control mode (any bit set in PSASUPER)

TCB TCB mode

TYP1 type 1 SVC in control

Note: The specification of LLOC, LOCK, or ALL automatically includes the cross memory local lock (CML).

ANY
Any mode you specify must occur to satisfy the match test, except when you specify HOME along with other modes and ANY. In that case, the unit of work must have been executing in the home address space when the event occurred and at least one of the other specified modes must occur to satisfy the match test

EVERY
Every mode you specify must occur to satisfy the match test

NUCEP | NUCMOD | N
The event must occur within a load module in the nucleus to satisfy the match test.

- If you do not specify *start* and *end*, SLIP monitors the entire module.
- If you specify only *start*, SLIP monitors a 1-byte range from the offset *start*.
- If you specify both *start* and *end*, SLIP monitors the range between the offset *start* and the offset *end*.

mod_name
The module name

value: 1-8 alphanumeric characters

start
The offset into the module (1-byte range)

value: 1-6 hexadecimal digits

start,*end*
The starting and ending offsets into the module

value: 1-6 hexadecimal digits for *start* and *end*. (Specify a value for *end* that is greater than or equal to that of *start*)

- For more information on choosing the *start* and *end* offsets relative to the environment, see *z/OS Problem Management*.

PRCNTLIM | PL
The percentage limit of system processing that is devoted to monitoring PER traps. At least 33.55 seconds must have elapsed since the first PER interruption before SLIP disables a trap because of this limit. If you do not specify PRCNTLIM for a non-ignore PER trap, SLIP assumes 10%.

When the processing limit is surpassed, SLIP disables the non-ignore enabled PER trap and issues message IEA411I. If you specified ACTION=TRDUMP, SLIP schedules an SVC dump.

p The percentage limit

value: an integer in the range 1-99. (The value SLIP computes to test

PRCNTLIM is an approximation. In addition, SLIP truncates the computed value to an integer before making the test.)

Note: Use caution in specifying a percentage limit of 99 because SLIP will not do percent limit checking.

PSWASC | PA

The PSW address space control mode, or modes, the system must be in to satisfy the match test.

asc_mode

The address space control (ASC) mode

value:

HOME or H
home ASC mode

PRIMARY or P
primary ASC mode

SECONDARY or S
secondary ASC mode

AR access register ASC mode

PVTEP

The event must occur within a private area load module, relative to the specified entry point, to satisfy the match test.

- If you do not specify *start* and *end*, SLIP monitors a range from the specified entry point to the end of the module.
- If you specify only *start*, SLIP monitors a 1-byte range from the offset *start*.
- If you specify both *start* and *end*, SLIP monitors the range between the offset *start* and the offset *end*.

If you specify PVTEP for an error trap, certain conditions cause the match test to fail when executing in non-task mode.

- local lock not held or obtained prior to the search of the CDE chain
- MEMTERM error
- DAT error

In addition,

- for non-PER and Storage Alteration PER traps, SLIP searches the primary address space prior to searching the home address space for the module. If the module is not executing in either address space, the match test fails.

mod_ep

The entry point name

value:

1. 1-8 alphanumeric characters
2. 1-7 alphanumeric characters appended with an asterisk. (SLIP interprets the asterisk as X'C0'.)

start

The offset into the module from the entry point (1-byte range)

value: 1-6 hexadecimal digits

*start***,***end*

The starting and ending offsets into the module

value: 1-6 hexadecimal digits for *start* and *end*. (Specify a value for *end* that is greater than or equal to that of *start*.)

PVTMOD | P

The event must occur within a private area load module to satisfy the match test.

- If you do not specify *start* and *end*, SLIP monitors the entire module.
- If you specify only *start*, SLIP monitors a 1-byte range from the offset *start*.
- If you specify both *start* and *end*, SLIP monitors the range between the offset *start* and the offset *end*.

If you specify PVTMOD for an error trap, certain conditions cause the match test to fail when executing in non-task mode.

- local lock not held or obtained prior to the search of the CDE chain
- MEMTERM error
- DAT error

In addition,

- for non-PER and Storage Alteration PER traps, SLIP searches the primary address space prior to searching the home address space for the module. If the module is not executing in either address space, the match test fails.

mod_name

The module name

value:

1. 1-8 alphanumeric characters
2. 1-7 alphanumeric characters appended with an asterisk. (SLIP interprets the asterisk as X'C0'.)

start

The offset into the module (1-byte range)

value: 1-6 hexadecimal digits

start,*end*

The starting and ending offsets into the module

value: 1-6 hexadecimal digits for *start* and *end*. (Specify a value for *end* that is greater than or equal to that of *start*.)

- For more information on choosing the *start* and *end* offsets relative to the environment, see *z/OS Problem Management*.

RANGE | RA

The event must occur at a virtual address, or within a range of virtual addresses, to satisfy the match test.

start

A virtual address (1-byte range)

value: 1-8 hexadecimal digits

start,*end*

A virtual address range

value: 1-8 hexadecimal digits for *start* and *end*. (SLIP makes no test to ensure that *end* is greater than *start*, because the specification of a starting address greater than an ending address causes the addresses to *wrap*.)

RBLEVEL | RB

Obtain the registers to resolve indirect addresses and the PSW used by

LPAMOD, PVTMOD, ADDRESS, and MODE from a particular RB. (RBLEVEL applies only to unlocked task mode errors.) If SLIP cannot find the RB specified by RBLEVEL, a no-match condition for the trap exists.

ERROR
Obtain the PSW from the RB prior to the SVC 13 (ABEND) RB (RB2 in Figure 1). SLIP obtains the registers from the SVC 13 RB (RB1 in Figure 1).

PREVIOUS
Obtain the PSW and registers from one RB prior to the one used in ERROR. (PSW from RB3 in Figure 1; registers from RB2 in Figure 1.)

NOTSVRB
Obtain the PSW from the most recent non-SVRB; the registers from the associated SVRB. (For example in Figure 1, if RB1, RB2, and RB3 are SVRBs, SLIP obtains the PSW from RB4 and the registers from RB3.)

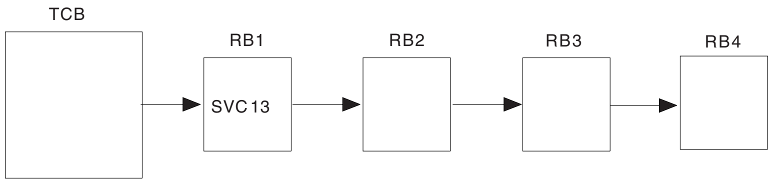


Figure 1. RB Structure

SA
Monitor a storage alteration PER trap

SB
Monitor a successful branch PER trap

For successful branch monitoring, PER processing does not check the address range specified on the RANGE, LPAMOD/EP, NUCMOD/EP, and PVTMOD/EP parameters. Therefore, any branch taken would cause a successful branch PER interrupt.

To prevent the foregoing interrupts, SLIP initially sets up instruction fetch monitoring for the specified address range. When an instruction fetch PER interrupt occurs, SLIP automatically switches PER monitoring to successful branch mode. However, if the instruction that caused the instruction fetch interrupt was a branch into the range, the trap does not match for that instruction.

If you specify ACTION=IGNORE for a successful branch PER trap, SLIP does not do mode switching; that is, the mode remains instruction fetch.

If an EXECUTE instruction has a successful branch target, the location of the EXECUTE instruction, not that of the executed branch, determines whether the branch was within the monitored range.

SET
Define a SLIP trap. If you do not specify IF, SA, or SB, you define a non-PER trap (error trap).

Example 1

Operation: Setting a SLIP trap using the instruction fetch PER event.

```
sl set,if,en,action=svcd,range=(cd3100),end
```

Example 2

Operation: Setting a SLIP trap using the storage alteration PER event.

```
sl set,sa,en,action=svcd,range=(cd3010,cd3013),
data=(cd3010,eq,00000000),end
```

Example 3

Operation: Setting a SLIP trap to obtain a dump with queue elements and control blocks.

```
sl set,id=50cx,a=svcd,comp=x30,errtyp=abend,jspgm=comrtn,
sdata=(sqa,rgn,trt,sum),end
```

- Setting a SLIP Error (non-PER) Trap

Syntax

```
{SLIP}            SET,SA ,{ACTION}={IGNORE                                                        }[,RECORD]
{SL  }                    {A     } {NODUMP                                                        }
                                   {NOSUP                                                         }
                                   {(NOSVCD[,NOSYSA][,NOSYSM][,NOSYSU])                           }
                                   {RECORD                                                        }
                                   {                                                              }
                                   {TRACE[,{TRDATA}=({STD},[REGS]    }[,[asid.]start,end])]       }
                                   {       {TD    }  {REGS           }                            }
                                   {                 {asid.}start,end}                            }
                                   {                                                              }
                                   {SVCD                                                          }
                                   {                                                              }
                                   {TRDUMP[,{TRDATA}=({STD}[,REGS]    }[,[asid.]start,end)]]      }
                                   {        {TD    }  {REGS           }                           }
                                   {                  {asid.}start,end}                           }
                                   {                                                              }
                                   {      ,{SDATA}=(option[,option]...)                           }
                                   {       {SD   }                                                }
                                   {                                                              }
                                   {      [,{ASIDLST}=(n[,n]...)]                                 }
                                   {      [ {AL     }           ]                                 }
                                   {                                                              }
                                   {      [,{LIST}=({asid.}start,end[,[asid.]start,end]...)]      }
                                   {      [ {LS  }                                         ]      }
                                   {                                                              }
                                   {      [,{SUMLIST}=({asid.}start,end[,[asid.]start,end]...)]   }
                                   {      [ {SL     }                                         ]   }
                                   {                                                              }
                                   {      [,DSPNAME=(asid.name[,asid.name]...)]                   }

                            [,ADDRESS=(start[,end])              ]
                            [                                    ]
                            [,{LPAEP   }=(mod-name{,start{,end}}) ]
                            [ {        }                          ]
                            [ {{LPAMOD}}                          ]
                            [ {{L     }}                          ]
                            [ {        }                          ]
                            [ {NUCEP   }                          ]
                            [ {        }                          ]
                            [ {{NUCMOD}}                          ]
                            [ {{N     }}                          ]
                            [ {        }                          ]
                            [ {{PVTEP }}                          ]
                            [ {        }                          ]
                            [ {{PVTMOD}}                          ]
                            [ {{P     }}                          ]

                            [,{ASID}=(id[,id...])]
                            [ {AS  }             ]

                            [,{DATA}=([(]...d-comp)[{{ ,     }{OR } {       }  }d-comp[)]]...)]
                            [ {DA  }               [{{)[)]...}{AND} {([(]...}  }              ]
                            [                      [{                          }              ]
                            [                      [{{)[,]   } { | } {[,]    } }              ]
                            [                      [{{)[)]...} { & } {([(]...} }              ]

                            [,{COMP}={hhh }[,{REASON}=code]]
                            [ {C   } {Uddd}[ {RE    }     ]]
```

```
[,DEBUG]

,{{ENABLE }}
 {{EN     }}
 {        }
 {{DISABLE}}
 {{D      }}

{,{ERRTYP}=({ALL           })}
{ {ER    } ({type[,type]...})}

{,ID=xxxx}

[,{JOBNAME}=j-name]
[ {J      }       ]

[,{JSPGM}=js-name]
[ {JS   }        ]

{,{MATCHLIM}=n}
{ {ML      }  }

{,{MODE}=({ALL            },{ANY   })}
{ {M   } ({cond[,cond]...)} {EVERY })}

[,PSWASC=(mode[,mode]...)]

{,{RBLEVEL}={ERROR   } }
{ {RB     } {PREVIOUS} }
{           {NOTSVRB } }

,{END}
 {E  }
```

Notes for Defining a SLIP Error (non-PER) Trap

Note:

1. Refer to the explanation of ID for the default SLIP uses.
2. Refer to the explanation of MATCHLIM for the defaults SLIP uses.
3. Refer to the explanation of the associated parameters for the defaults SLIP uses.
4. Refer to the explanation of the associated parameters for the default SLIP uses.
5. Refer to the explanation of SDATA for the defaults SLIP uses.
6. SET and END are positional parameters; all others are keyword parameters.
7. You may specify NOSVCD, NOSYSA, NOSYSM, or NOSYSU in any order; for example:

   ```
   ACTION=(NOSYSU)
   ACTION=(NOSYSA,NOSVCD)
   ACTION=(NOSYSM,NOSYSU,NOSYSA,NOSVCD)
   ```

   are all valid specifications.
8. In the DATA parameter, the elements of *data-compare* are:

   [*asid.*]*target*[*(b)*],*operator*[{A|C}[(*n*)]],*preval*
9. In the DATA parameter, a maximum of 16 levels of parentheses are allowed; that is, no more than 16 unmatched left parentheses may appear in a DATA parameter specification.

- Setting a PER Trap for Instruction Fetch or a Successful Branch

```
{SLIP}            SET,{IF} ,{ACTION}={(IGNORE[,RECOVERY])                                                 }
{SL  }                {SB}  {A     } {STDUMP                                                              }
                                     {STRACE                                                              }
                                     {RECOVERY                                                            }
                                     {                                                                    }
                                     {TRACE[,RECOVERY][,{TRDATA}=({STD},[REGS]    }[,[asid.]start,end)]]  }
                                     {                  {TD    }  {REGS           }                       }
                                     {                            {asid.}start,end}                       }
                                     {                                                                    }
                                     {(SVCD[,RECOVERY])                                                   }
                                     {SYNCSVCD                                                            }
                                     {                                                                    }
                                     {TRDUMP[,RECOVERY][,{TRDATA}=({STD}[,REGS]    }[,[asid.]start,end)]] }
                                     {                   {TD    }  {REGS           }                      }
                                     {                             {asid.}start,end}                      }
                                     {                                                                    }
                                     {     ,{SDATA}=(option[,option]...)                                  }
                                     {      {SD   }                                                       }
                                     {                                                                    }
                                     {     [,{ASIDLST}=(n[,n]...)]                                        }
                                     {     [ {AL     }           ]                                        }
                                     {                                                                    }
                                     {     [,{LIST}=({asid.}start,end[,[asid.]start,end]...)]             }
                                     {     [ {LS  }                                         ]             }
                                     {                                                                    }
                                     {     [,{SUMLIST}=({asid.}start,end[,[asid.]start,end]...)]          }
                                     {     [ {SL     }                                         ]          }
                                     {                                                                    }
                                     {     [,DSPNAME=(asid.name[,asid.name]...)]
                                     }

                          [,RANGE=(start[,end])                 ]
                          [                                     ]
                          [,{LPAEP   }=(mod-name{,start{,end}}) ]
                          [ {        }                          ]
                          [ {{LPAMOD}}                          ]
                          [ {{L     }}                          ]
                          [ {        }                          ]
                          [ {NUCEP   }                          ]
                          [ {        }                          ]
                          [ {{NUCMOD}}                          ]
                          [ {{N     }}                          ]
                          [ {        }                          ]
                          [ {{PVTEP }}                          ]
                          [ {        }                          ]
                          [ {{PVTMOD}}                          ]
                          [ {{P     }}                          ]

                          [,{ASID}=(id[,id...])]
                          [ {AS  }             ]

                          [,{ASIDLST}=(n[,n]...)]
                          [ {AL     }           ]

                          [,{DATA}=([(]...d-comp)[{{ ,     }{OR } {       }  }d-comp[)]]...)]
                          [ {DA  }               [{{)[)]...}{AND} {([(]...}  }              ]
                          [                      [{                          }              ]
                          [                      [{{)[,]   } { | } {[,]    } }              ]
                          [                      [{{)[)]...} { & } {([(]...} }              ]
```

```
[,DEBUG]

,{{ENABLE }}
 {{EN     }}
 {        }
 {{DISABLE}}
 {{D      }}

{,ID=xxxx}

[,{JOBNAME}=j-name]
[ {J      }       ]

[,{JSPGM}=js-name]
[ {JS   }        ]

{,{MATCHLIM}=n}
{ {ML      }  }

{,{MODE}=({ALL           },{ANY   }))}
{ {M   } ({cond[,cond...]} {EVERY }))}

,{PRCNTLIM}=p
 {PL      }

[,PSWASC=(mode[,mode]...)]

[,RANGE=(start[,end])]

,{END}
 {E  }
```

Notes® for Defining a SLIP PER Trap for Instruction Fetch or Successful Branch

Note:
1. Refer to the explanation of ID for the default SLIP uses.
2. Refer to the explanation of PRCNTLIM for the default SLIP uses.
3. Refer to the explanations of MATCHLIM and STRACE for the defaults SLIP uses.
4. Refer to the explanations of MATCHLIM, STDUMP, and STRACE for the defaults SLIP uses.
5. Refer to the explanation of the associated parameters for the defaults SLIP uses.
6. Refer to the explanation of the associated parameters for the default SLIP uses.
7. Refer to the explanation of SDATA for the defaults SLIP uses.
8. SET, IF, SB, and END are positional parameters; all others are keyword parameters.
9. Enable only one non-ignore PER trap at any one time. If you attempt to set an enabled non-ignore trap while one is already enabled, SLIP defines the trap, forces it to the disabled state, and issues message IEE740I. If you attempt to enable a non-ignore PER trap while one is already enabled, SLIP denies the request and issues message IEE741I.
10. If you do not specify RECOVERY in conjunction with another parameter, the use of the indicated parentheses is optional.
11. In the DATA parameter, the elements of *data-compare* are:

    [*asid.*]*target*[*(b)*],*operator*[{A|C}[(*n*)]],*preval*

12. In the DATA parameter, a maximum of 16 levels of parentheses are allowed; that is, no more than 16 unmatched left parentheses may appear in a DATA parameter specification.

- Setting a PER Trap for Storage Alteration

```
{SLIP}            SET, SA  ,{ACTION}={(IGNORE[,RECOVERY])                                                }
{SL  }                      {A     } {STDUMP                                                             }
                                     {STRACE                                                             }
                                     {RECOVERY                                                           }
                                     {                                                                   }
                                     {TRACE[,RECOVERY][,{TRDATA}=({STD},[REGS]    }[,[asid.]start,end)]] }
                                     {                  {TD    }  {REGS           }                      }
                                     {                            {asid.}start,end}                      }
                                     {                                                                   }
                                     {(SVCD[,RECOVERY])                                                  }
                                     {SYNCSVCD                                                           }
                                     {                                                                   }
                                     {TRDUMP[,RECOVERY][,{TRDATA}=({STD}[,REGS]    }[,[asid.]start,end)]]}
                                     {                   {TD    }  {REGS           }                     }
                                     {                             {asid.}start,end}                     }
                                     {                                                                   }
                                     {      ,{SDATA}=(option[,option]...)                                }
                                     {       {SD   }                                                     }
                                     {                                                                   }
                                     {      [,{ASIDLST}=(n[,n]...)]                                      }
                                     {      [ {AL     }           ]                                      }
                                     {                                                                   }
                                     {      [,{LIST}=({asid.}start,end[,[asid.]start,end]...)]           }
                                     {      [ {LS  }                                          ]          }
                                     {                                                                   }
                                     {      [,{SUMLIST}=({asid.}start,end[,[asid.]start,end]...)]        }
                                     {      [ {SL     }                                          ]       }
                                     {                                                                   }
                                     {      [,DSPNAME=(asid.name[,asid.name]...)]                        }

                          [,ADDRESS=(start[,end])               ]
                          [                                     ]
                          [,{LPAEP   }=(mod-name{,start{,end}}) ]
                          [ {        }                          ]
                          [ {{LPAMOD}}                          ]
                          [ {{L     }}                          ]
                          [ {        }                          ]
                          [ {NUCEP   }                          ]
                          [ {        }                          ]
                          [ {{NUCMOD}}                          ]
                          [ {{N     }}                          ]
                          [ {        }                          ]
                          [ {{PVTEP }}                          ]
                          [ {        }                          ]
                          [ {{PVTMOD}}                          ]
                          [ {{P     }}                          ]

                          [,{ASID}=(id[,id...])]
                          [ {AS  }             ]

                          [,{ASIDLST}=(n[,n]...)]
                          [ {AL     }           ]

                          [,{DATA}=([(]...d-comp)[{{ ,     }{OR } {       }  }d-comp[)]]...)]
                          [ {DA  }               [{{)[)]...}{AND} {([(]...}  }              ]
                          [                      [{                          }              ]
                          [                      [{{)[,]   } { | } {[,]    } }              ]
                          [                      [{{)[)]...} { & } {([(]...} }              ]
```

```
[,DEBUG]

[,DSSA=(asid.name[,asid.name]...)]

,{{ENABLE }}
 {{EN     }}
 {        }
 {{DISABLE}}
 {{D      }}

{,ID=xxxx}

[,{JOBNAME}=j-name]
[ {J      }       ]

[,{JSPGM}=js-name]
[ {JS   }        ]

{,{MATCHLIM}=n}
{ {ML      }  }

{,{MODE}=({ALL           },{ANY   }))}
{ {M   } ({cond[,cond]...)} {EVERY }))}

,{PRCNTLIM}=p}
 {PL      }  }

[,PSWASC=(mode[,mode]...)]

[,RANGE=(start[,end])]

,{END}
 {E  }
```

Notes for Defining a SLIP PER Trap for Storage Alteration

Note:

1. Refer to the explanation of ID for the default SLIP uses.
2. Refer to the explanation of PRCNTLIM for the default SLIP uses.
3. Refer to the explanation of MATCHLIM and STRACE for the defaults SLIP uses.
4. Refer to the explanation of MATCHLIM, STDUMP, and STRACE for the defaults SLIP uses.
5. Refer to the explanation of the associated parameters for the defaults SLIP uses.
6. Refer to the explanation of the associated parameters for the default SLIP uses.
7. Refer to the explanation of SDATA for the defaults SLIP uses.
8. *name* is required unless you specify the identifier SA on a storage alteration trap. In that case, SLIP dumps the space of the storage being altered.
9. SET, SA, and END are positional parameters; all others are keyword parameters.
10. RANGE and IGNORE are mutually exclusive parameters.
11. Enable only one non-ignore PER trap at any one time. If you attempt to set an enabled non-ignore trap while one is already enabled, SLIP defines the trap, forces it to the disabled state, and issues message IEE740I. If you attempt to enable a non-ignore PER trap while one is already enabled, SLIP denies the request and issues message IEE741I.
12. If you do not specify RECOVERY in conjunction with another parameter, the use of the indicated parentheses is optional.
13. In the DATA parameter, the elements of *data-compare* are:

[*asid.*]*target*[*(b)*],*operator*[{A|C}[(*n*)]],*preval*

14. In the DATA parameter, a maximum of 16 levels of parentheses are allowed; that is, no more than 16 unmatched left parentheses may appear in a DATA parameter specification.

- Deleting Previously Defined Traps

```
               ┌ , ALL    ┐
SL[ IP]  DEL ──┤          ├
               └ , ID=xxxx┘
```

- All parameters are positional parameters.
- If more than one user of SLIP defines traps in the system, coordinate their actions in deleting traps to prevent undesirable results. For example, you can unknowingly delete traps previously defined by another user by issuing the following command:

  ```
  slip del,all
  ```

  Before issuing a global SLIP DEL command, issue the DISPLAY SLIP subcommand to find out the status of other SLIP traps.

  To prevent undesirable results, delete SLIP traps explicitly using:

  ```
  id=specific-trap-identifier
  ```
- When a trap is deleted by a TSO/E user ID other than the one who set the trap, SLIP notifies the originator of the trap of its changing status and the user ID responsible by issuing message IEE727I.

DEL
Delete an existing trap or traps. (If TRDUMP is active, SLIP schedules an SVC dump.)

ALL
Delete all traps

ID Delete a trap

xxxx
Trap identifier

value: 1-4 alphanumeric characters

Example

Operation: Delete a previously defined trap.

```
sl del,id=50cb
```

- Enabling or Disabling Previously Defined Traps

```
               ┌ EN[ ABLE] ┐ ┌ , ALL    ┐
SL[ IP]  MOD, ─┤           ├─┤          ├
               └ D[ ISABLE]┘ └ , ID=xxxx┘
```

- All parameters are positional parameters.
- If more than one user of SLIP defines traps in the system, coordinate their actions in enabling and disabling traps to prevent undesirable results. For

example, you can unknowingly enable traps previously disabled by another user by issuing one of the following commands:

```
slip mod, en, all       }  equivalent specifications
slip mod, en, id=****   }  enable all traps

slip mod, en, id=55**      enable all traps whose ids
                           begin with 55 and are four
                           alphameric characters in length
```

Before issuing global enable or disable SLIP MOD commands, issue the DISPLAY SLIP subcommand to find out the status of other SLIP traps.

To prevent undesirable results, enable and disable SLIP traps explicitly using:

`id=specific-trap-identifier`

- When a trap is enabled or disabled by a TSO/E user ID other than the one who set the trap, SLIP notifies the originator of the trap of its changing status and the user ID responsible by issuing message IEE727I.

MOD
Change the status of an existing trap or traps

ENABLE | EN
Enable a previously defined trap, or traps; or enable a disabled trap, or traps

DISABLE | D
Disable an enabled trap or traps. (If TRDUMP is active, SLIP schedules an SVC dump.)

ALL
Change the status of *all* traps.

ID Change the status of *a* trap, a set of traps, or all traps

xxxx
Trap identifier(s)

value:

1. 1-4 alphanumeric characters (a specific trap)
2. 1-3 alphanumeric characters and 1-3 occurrences of an asterisk (*) (a set of traps)
3. **** (all traps)

Example 1

Operation: Enable a disabled trap.

```
sl mod,en,id=6a
```

Example 2

Operation: Enable all disabled traps whose ids begin with the letter A and end with the digit 0 and are four alphanumeric characters in length

```
sl mod,en,id=a**0
```

Example 3

Operation: Enable previously defined traps with ids in the ranges of 50-59 and 5A-5Z; and those with specific ids of 5$, 5#, and 5@

```
sl mod,en,id=5*
```

OPERATOR—STOPMN subcommand

Use the STOPMN subcommand to terminate the monitoring operations of the MONITOR subcommand. This subcommand halts the display of information at your terminal.

The syntax of the STOPMN subcommand of OPERATOR is:

```
{STOPMN}  {JOBNAMES}
{      }  {SESS    }
{PM    }  {STATUS  }
```

JOBNAMES
Stop the display of the names of jobs as they start and terminate

SESS
Stop the display of TSO/E user IDs as users initiate and terminate terminal sessions

STATUS
Stop the display of the names and volume serial numbers of data sets with dispositions of KEEP, CATLG, or UNCATLG when the data sets are freed

Example 1

Operation: Stop the display of the names of jobs as they start and terminate.

```
stopmn jobnames
```

Example 2

Operation: Stop the display of TSO/E user IDs as terminal sessions are initiated and terminated.

```
pm    sess
```

PARMLIB command

Use the PARMLIB command to:
- Display the specifications in the active IKJTSOxx member of SYS1.PARMLIB for:
  - one system
  - all systems in a sysplex, or
  - a subset of systems in a sysplex
- Dynamically change the active member without a re-IPL for:
  - one system
  - all systems in a sysplex, or
  - a subset of systems in a sysplex
- Check the syntax of any IKJTSOxx member of SYS1.PARMLIB.

Note: You can also place IKJTSOxx members in data sets other than SYS1.PARMLIB. This extra flexibility allows you, for example, to separate your data from IBM supplied data. You can specify a list of PARMLIB data sets that will comprise a logical concatenation (logical PARMLIB) for the life of the MVS system (similar to LPALST and LNKLST for LPA and link-list libraries).

The syntax of the PARMLIB command is:

```
PARMLIB  { { LIST(ALL)                        } { ROUTE ( } { *          } ) } }
         { { LIST (statement-name)            } {         } { systemname }   } }
         { { UPDATE (member-name-suffix)      } {         } { groupname  }   } }
         {   CHECK (member-name-suffix)                                        }
```

- PARMLIB is an authorized command.
- If RACF is installed and active, you require either authority to the RACF PARMLIB resource of the RACF TSOAUTH resource class or authority through the installation exit routine, IKJPRMX1, to issue the command. (See *z/OS TSO/E Customization* for detailed information concerning IKJPRMX1.)
- If RACF is not installed or active, you require authority either through an equivalent security product or through the installation exit routine to issue the command.

LIST

Display specifications in the active IKJTSOxx member of SYS1.PARMLIB.

The output of the PARMLIB LIST command also shows the system that created the PARMLIB member, the timestamp and the user id of the person who issued the PARMLIB UPDATE command.

In sysplex systems this command can be used to display the PARMLIB settings on one, all, or a subset of the systems in the sysplex by specifying the ROUTE operand (see 106). In sysplex systems, the responses are gathered together and grouped so that all systems which have the same settings are shown in one display.

```
TSO/E PARMLIB SETTINGS:

   SYS2.PARMLIB(IKJTSOWS) ON VOLUME vvvvvv
   Activated by user uuuuuuu on YYYY-MM-DD at HH:MM:SS from system ssssssss
   Applies to        TSOA     TSOB

   SYS1.PARMLIB(IKJTSOFF) ON VOLUME vvvvvv
   Activated by user uuuuuuu on YYYY-MM-DD at HH:MM:SS from system ssssssss
   Applies to        TSOC

         CURRENT SETTINGS FOR CONSOLE:

         INITUNUM             1000
         INITSNUM             1000
         MAXUNUM             10000
         MAXSNUM             10000

   ----------------------------------------------------------------------------

   SYS1.PARMLIB(IKJTSOAB) on volume vvvvvv
   Activated by user STCUSER on YYYY-MM-DD at HH:MM:SS from system TSOE
   Applies to        TSOE

         CURRENT SETTINGS FOR CONSOLE:

         INITUNUM             1000
         INITSNUM             1000
         MAXUNUM             20000
         MAXSNUM             20000
```

Figure 2. Sample PARMLIB LIST Output

ALL
Display all the specifications in the active IKJTSOxx member of SYS1.PARMLIB

statement_name
Only display the specifications from an individual statement in the active IKJTSOxx member of SYS1.PARMLIB

value: as indicated in the following list:

ALLOCATE
ALLOCATE command default data set status

AUTHCMD
list of authorized commands

AUTHPGM
list of programs that are authorized when invoked via the CALL command

AUTHTSF
list of programs that are authorized when invoked through the TSO/E service facility

CONSOLE
message processing defaults for the CONSOLE command and its services

HELP list of help data sets for different languages

LOGON
list of options for LOGON processing

NOTBKGND
list of commands not supported in the background

PLATCMD
list of commands that can run on the TSO/E command invocation platform

PLATPGM
list of programs that can run on the TSO/E command invocation platform when invoked through TSO/E service facility

SEND SEND, OPERATOR SEND, and LISTBC command defaults and active broadcast data set information

TEST list of additional commands and subcommands valid under TEST and TESTAUTH

TRANSREC
TRANSMIT/RECEIVE command options and defaults

For detailed information concerning each of the statements in the IKJTSOxx member of SYS1.PARMLIB, see *z/OS MVS Initialization and Tuning Reference*.

UPDATE
Dynamically change, without a re-IPL, the active IKJTSOxx member of SYS1.PARMLIB.

In sysplex systems this command can be used to update one, all, or a subset of the systems in the sysplex by specifying the ROUTE operand (see 106).

When updating sysplex systems, every symbolic variable in the specified IKJTSOxx member of SYS1.PARMLIB is resolved on the system where the command is issued and run and then the resulting information is passed on to the systems as specified by the ROUTE operand (see 106).

When updating sysplex systems, if the command fails on any of the systems, no backout is done on the systems that completed the PARMLIB UPDATE successfully.

Before you change the active IKJTSOxx member using the PARMLIB UPDATE command, it is highly recommended that you check the syntax of the member using the PARMLIB CHECK command (see below).

See *z/OS TSO/E Customization* for detailed information on using SYS1.PARMLIB versus CSECTS IKJEFTE2, IKJEFTE8, IKJEFTAB, and IKJEFTNS to maintain the lists of authorized commands and programs, and commands not supported in the background.

member_name_suffix
Identification of the now active IKJTSOxx member of SYS1.PARMLIB

value: a maximum of two alphabetic and/or numeric characters

The system appends the two characters to IKJTSO to identify the specified member. You are responsible for ensuring that the member exists in parmlib.

ROUTE
In a sysplex, the ROUTE operand causes the PARMLIB command processor to gather information from (LIST operand) or send information to (UPDATE operand) the specified systems in the sysplex.

When the LIST operand is specified, the PARMLIB command processor gathers information from the specified systems and presents the information on the system where the PARMLIB command is issued.

When the UPDATE operand is specified, the PARMLIB command processor resolves every symbolic variable in the specified member and then sends identical information to each of the specified systems. If no broadcast data set switch is detected on the system where the command is issued, no prompt for confirmation will occur, even if the PARMLIB UPDATE results in a broadcast data set switch on one or more systems in the sysplex. The PROMPT/NOPROMPT keyword has no effect beyond the system where the command is issued. If a separate prompt for each system is needed, consider using the MVS ROUTE *ALL,SET IKJTSO=xx command instead.

`*` All systems in the sysplex.

`systemname`
Only one system in the sysplex.

`groupname`
Only the named subset of systems in the sysplex.

Systemname and *groupname* are in MVS name token format. For defining them, use the MVS program IEEGSYS. For more information about IEEGSYS, see *z/OS MVS Planning: Operations*.

`CHECK`
Check the syntax of any IKJTSOxx member of SYS1.PARMLIB

`member_name_suffix`
Identification of the IKJTSOxx member of SYS1.PARMLIB

value: a maximum of two alphabetic and/or numeric characters

The system appends the two characters to IKJTSO to identify the specified member. You are responsible for ensuring that the member exists in parmlib.

PARMLIB command return codes

Table 5. PARMLIB command return codes

0	Successful completion.
4	Cleanup failed.
6	Check failed.
8	Update failed.
10	List failed.
12	User not authorized.
14	Authority was not verified.
16	Error encountered in parse.
20	Error setting up recovery.
24	Exit requested termination.
28	Error in router or exit.
32	Allocate of logical parmlib failed.
36	Free of logical parmlib failed.
40	Error in system command.

Table 5. PARMLIB command return codes (continued)

72	PARMLIB LIST(CONSOLE) failed. The level of MVS is not SP 4.1.0 or above.
76	Not running authorized.

Example 1

Operation: Display the table of authorized commands.

```
parmlib list(authcmd)
```

Example 2

Operation: Change the active IKJTSOxx member of SYS1.PARMLIB to IKJTSO03.

```
parmlib update(03)
```

Example 3

Operation: Display all the specifications in the active IKJTSOxx member of SYS1.PARMLIB.

```
parmlib
```

Example 4

Operation: Display the message processing defaults for the CONSOLE command and its services.

```
parmlib list(console)
```

Example 5

Operation: Check the syntax of the IKJTSOxx parmlib member IKJTSO03.

```
parmlib check(03)
```

Example 6

Example 6 is a sample output from the command: PARMLIB LIST(CONSOLE),ROUTE(*).

```
PARMLIB LIST(CONSOLE),ROUTE(*)

   SYS2.PARMLIB(IKJTSOWS) ON VOLUME vvvvvv
  Activated by user uuuuuuu on YYYY-MM-DD at HH:MM:SS from system ssssssss
  Applies to        TSOA     TSOB

  SYS1.PARMLIB(IKJTSOFF) ON VOLUME vvvvvv
  Activated by user uuuuu on YYYY-MM-DD at HH:MM:SS from system ssssssss
  Applies to        TSOC

         CURRENT SETTINGS FOR CONSOLE:

         INITUNUM             1000
         INITSNUM             1000
         MAXUNUM             10000
         MAXSNUM             10000

------------------------------------------------------------------

  SYS1.PARMLIB(IKJTSOAB) on volume vvvvvv
  Activated by user STCUSER on YYYY-MM-DD at HH:MM:SS from system TSOE
  Applies to        TSOC

         CURRENT SETTINGS FOR CONSOLE:
```

```
        INITUNUM             1000
        INITSNUM             1000
        MAXUNUM             20000
        MAXSNUM             20000
```

Example 7

Example 7 is a sample output from the command: PARMLIB LIST(SEND).

```
TSO/E PARMLIB SETTINGS:

  SYS1.PARMLIB(IKJTSOA1) on volume PRMVOL
  Activated by **IPL** on 2001-10-29 at 11:55:48 from system OSV313
  Applies to:       OSV313

        THE FOLLOWING ARE THE PARMLIB OPTIONS FOR SEND:

        OVERSEND (ON)
        USERSEND (ON)
        SAVE (ON)
        CHKBROD (OFF)
        LOGNAME(*)
        USEBROD (ON)
        MSGPROTECT (OFF)
        SYSPLEXSHR (OFF)
        OPERSEWAIT (ON)
        USERLOGSIZE(1,2)
        BROADCAST(DATASET(SYS2.BROADCAST)
                      VOLUME(BRDVOL) TIMEOUT(10) PROMPT)
```

Note: LOGNAME(*) indicates that messages (mail) are to be stored in the broadcast data set.

RACONVRT command

Use the RACONVRT command as an aid in converting from SYS1.UADS to the RACF data base. During the conversion process, RACONVRT does not migrate the TSO command that was specified on the previous logon. Therefore, the command field in the logon panel contains no data the first time the user logs on after the conversion is complete. If the user specifies a command in the TSO command field on the logon panel, TSO/E saves that command for the next logon. For detailed information about the command, see *z/OS TSO/E Customization*.

The syntax of the RACONVRT command is:

```
           ┌  ALL                                                                      ┐
           │                                                                           │
RACONVRT  <   ┌INCLUDE┐   ┌userid[ , userid]...                                      ┐  >
           │ <        > ( <userid:userid[ , userid: userid]...                       > )│
           └  └EXCLUDE┘   └userid[ , userid]..., userid: userid[ , userid: userid]...┘  ┘
```

- All parameters are keyword parameters.
- RACONVRT is an authorized command.
- If RACF is installed and active, you require RACF SPECIAL authority to issue the command.
- If RACF is not installed or active, you require ACCOUNT authority to issue the command.

`ALL`
Convert all entries in SYS1.UADS to the RACF data base

When converting a large number of entries, the system may not be able to obtain enough storage to process all entries. RACONVRT processing then terminates with a message. It is, therefore, recommended to issue several RACONVRT commands with the INCLUDE/ EXCLUDE parameters whenever a large number of entries are to be converted.

`INCLUDE`
Convert the specified entries (user IDs) in SYS1.UADS to the RACF data base

`EXCLUDE`
Do not convert the specified entries (user IDs) in SYS1.UADS to the RACF data base; however, convert all those not specified

`userid`
The user ID of an existing entry in SYS1.UADS that you are converting to the RACF data base

`userid``:`*`userid`*
A range of user IDs of existing entries in SYS1.UADS that you are converting to the RACF data base

value:

1. 1-7 alphanumeric characters, beginning with an alphabetic or special character
2. 1-6 alphanumeric characters, beginning with an alphabetic or special character and appended with an asterisk (*) - a set of user IDs

If a user ID has not logged on and off since changes were made to its SYS1.UADS entry, RACONVRT will not include an account number or procedure in the ADDUSER or ALTUSER command for that user ID. When a user ID logs off, the default account number and procedure are saved in SYS1.UADS. RACONVRT uses these defaults when creating the ADDUSER or ALTUSER command to add the user ID to RACF. If the user ID has not logged on and off since changes were made to SYS1.UADS, then defaults do not exist for the user ID in SYS1.UADS. This causes RACONVRT to create an ADDUSER or ALTUSER command without the ACCT or PROC operands.

RACONVRT command return codes

Table 6. RACONVRT command return codes

0	Processing completed successfully.
4	Processing completed unsuccessfully. The recovery environment could not be established.
8	Processing completed unsuccessfully. The command was not invoked in an authorized state.
12	Processing completed unsuccessfully. The invoker has insufficient authority to issue the command.
16	Processing completed unsuccessfully. The command operand could not be parsed.
20	Processing completed unsuccessfully. The I/O routines, IKJRUR04 and IKJEFA51, could not be loaded.

Table 6. RACONVRT command return codes (continued)

24	Processing completed unsuccessfully. Storage for the necessary tables could not be obtained.
28	Processing completed unsuccessfully. The SYS1.UADS data set could not be opened.
32	Processing completed unsuccessfully. An I/O error occurred while reading the SYS1.UADS data set.
36	Processing completed unsuccessfully. RACF is not active.
40	Processing completed unsuccessfully. An error occurred during RACF processing.
44	Processing completed unsuccessfully. PUTGET failed when prompting the user with message IKJ56781A.
48	Processing completed unsuccessfully. The user has terminated RACONVRT.
52	Processing completed unsuccessfully. The CLIST data set is allocated with attributes that are not valid.
56	Processing completed unsuccessfully. The CLIST data set could not be allocated.
60	Processing completed unsuccessfully. The CLIST data set could not be opened.
64	Processing completed unsuccessfully. An I/O error occurred writing to the CLIST data set.

RECEIVE command

The complete syntax and function of the RECEIVE command are described in *z/OS TSO/E Command Reference*. The parameters described here are intended for your use. You may use them, in conjunction with corresponding parameters on the TRANSMIT command, to test your exit routines and debug user-written control records. (See *z/OS TSO/E Customization* for detailed information on the exit routines and control records.)

The syntax of the RECEIVE command is:

```
                               ┌ ┌INDDNAME(ddname)┐ ┐
                               │ └INFILE(ddname)  ┘ │
RECEIVE  USERID(userid)       <                      > COPY
                               │ ┌INDSNAME(dsn)   ┐ │
                               └ └INDATASET(dsn)  ┘ ┘
```

You cannot explicitly specify COPY as a parameter on the RECEIVE command. COPY is one of the responses to the RECEIVE prompting message INMR909I.

- RECEIVE is an authorized command.

USERID

The specification of a user ID other than your own. (The specification of this parameter requires OPERATOR authority or authorization through the RECEIVE initialization exit (INMRZ01).

userid

Any user ID (The user ID may exist in SYS1.UADS or the RACF data base at the target node or may be a nonexistent user ID.)

COPY
Do not restore the transmitted data to its original format, but copy it 'as is' from the JES SPOOL. (The use of this operand allows you to examine the data in its transmitted form to debug problems when RECEIVE cannot process the transmitted data.)

INDDNAME | INFILE
A pre-allocated file used as the input data set to receive the transmitted data. Define the data set with RECFM=F, FB, V, VB, or U. For F and FB, LRECL=80. The remaining DCB attributes are installation options.

ddname
The name on a DD statement that identifies the data set. The data set may be sequential or partitioned but must be the same as the specification for OUTDDNAME or OUTFILE.

INDSNAME | INDATASET
A sequential data set used as the input data set to receive the transmitted data. Define the data set with RECFM=F, FB, V, VB, or U. For F and FB, LRECL=80. The remaining DCB attributes are installation options.

dsn
The name of a **sequential** data set.

value: a name that conforms to TSO/E data set naming conventions

SYNC command

Use the SYNC command to initialize the broadcast data set and synchronize it with either the UADS, the TSO/E segment of the RACF data base, or both.

When issued from a batch environment, the SYNC command accesses the broadcast data set via the SYSLBC DD name. If no SYSLBC DD statement is specified in the job, the active broadcast data set will be used by the command. If a SYSLBC DD statement is specified in the job, the referenced data set will be used as the broadcast data set.

TSO/E copies the user IDs from the UADS and/or the TSO/E segment of the RACF data base into the broadcast data set.

SYNC also formats the NOTICES section of the broadcast data set to reserve room for the maximum number of messages. (Use the IKJBCAST macro to specify the maximum number of messages.)

If you use SYNC when the broadcast data set exists, TSO/E deletes all MAIL from the data set.

In addition, if you use SYNC after you change the message limit for the NOTICES section and the broadcast data set exists (is initialized), the data set is cleared (all MAIL and NOTICES are deleted).

The syntax of the SYNC command is:

```
      ⎧ BOTH ⎫
SYNC  ⎨ RACF ⎬
      ⎩ UADS ⎭
```

- SYNC is an authorized command.
- You require ACCOUNT authority to issue the command.
- To synchronize the broadcast data set with the RACF data base, RACF must be installed and active.

BOTH
Synchronize the broadcast data set with both the TSO/E segment of the RACF data base and SYS1.UADS, provided that the SYS1.UADS data set was previously allocated to ddname SYSUADS. If it was not previously allocated, the broadcast data set is synchronized with the TSO/E segment of the RACF data base only.

RACF
Synchronize the broadcast data set only with the TSO/E segment of the RACF data base

UADS
Synchronize the broadcast data set only with SYS1.UADS

SYNC command return codes

Table 7. SYNC command return codes

0	Processing successful.
12	Processing unsuccessful. An error message has been issued.

TESTAUTH command

Use the TESTAUTH command to test an authorized program. Most of the functions of the TEST command that are available for testing an unauthorized program are also available through the TESTAUTH command for testing an authorized program. (See *z/OS TSO/E Command Reference* for detailed information.)

However, the TESTAUTH command *does not* support:
- the testing of a currently executing program
- the testing of object modules

In addition, do not use the TESTAUTH command to:
- modify storage that has a protection key of 0 through 7
- set breakpoints in storage that has a protection key of 0 through 7 (Although TESTAUTH allows you to set a breakpoint, the subsequent execution of a subcommand may give erroneous results.)

Restriction: TESTAUTH allows a user to test AMODE 24 or AMODE 31 programs. Testing programs with any other AMODE will have unpredictable results.

To use TESTAUTH to test a program, ensure that you load the program from an APF-authorized library. The system loads the program above or below 16 MB in virtual storage based on the program's RMODE attribute. The system uses the specified data set (PDS) as a TASKLIB for the program and initializes registers 2 through 12 to X'FFFFFFFF' to allow you to see which registers the program alters.

When you are testing a program, the program can invoke other load modules, if they are members of the same PDS. The services by which one member can invoke another in the same PDS include LINK, LOAD, XCTL, and ATTACH. If the program you are testing attempts to LOAD, LINK, XCTL, or ATTACH another module, the system uses the following search order sequence:
1. TASKLIB
2. STEPLIB
3. JOBLIB
4. LPA
5. LNKLST

If the module does not reside in any of those libraries, the system cannot find it. To avoid that situation, bring the module into virtual storage by using the LOAD subcommand of TESTAUTH.

All TESTAUTH subcommands are effective only in the HOME address space.

The syntax of the TESTAUTH command is:

```
                                                                  ┌ CP   ┐
TESTA[ UTH]  {'data-set-name' } ['parameter[ , parameter] . . . ']  ┤      ├  LOAD
                                                                  └ NOCP ┘
```

- TESTAUTH is an authorized command.
- If RACF is installed and active, you require either authority to the RACF TESTAUTH resource of the RACF TSOAUTH resource class or authority through the installation exit routine, IKJEGAUI, to issue the command. (See *z/OS TSO/E Customization* for detailed information concerning IKJEGAUI.)
- If RACF is not installed or active, you require authority either through an equivalent security product or through the installation exit routine to issue the command.
- Do not name a program you want to test
  - TEST
  - TESTAUTH
  - TESTA
  - the same as an existing TSO/E service routine.

data_set_name
: The name of the data set that contains the program. Enclose *data_set_name* in single quotation marks or TSO/E will fully qualify the data set name.

  value: a valid data set name

  parameter
  : Pass a parameter or list of parameters to the program. The maximum length allowed for the list of parameters is 100 characters, including delimiters.

    value: a value acceptable to that program

    CP
    : The program is a command processor

NOCP
The program is not a command processor

LOAD
The program is in load module format

A program in load module format has been processed by the linkage editor and is a member of a partitioned data set (PDS).

When using the TESTAUTH command, you can use:

1. any IBM-supplied TEST subcommand. (See *z/OS TSO/E Command Reference* for the complete syntax and description of all TEST subcommands.)
2. any command and subcommand you specify on the TEST statement in the IKJTSOxx member of SYS1.PARMLIB (See *z/OS MVS Initialization and Tuning Guide* for information concerning the IKJTSOxx member and *z/OS TSO/E Customization* for information on writing TEST subcommands.)

Example 1

Operation: Test the authorized program AUTHPGM that resides in SYS1.LINKLIB.

```
testauth 'sys1.linklib(authpgm)'
```

Example 2

Operation: Test the authorized command processor AUTHCMD that resides in SYS1.LINKLIB.

```
testauth 'sys1.linklib(authcmd)' cp
```

TRANSMIT command

The complete syntax and function of the TRANSMIT command are described in *z/OS TSO/E Command Reference*. The parameters described here are intended for your use. You may use them, in conjunction with corresponding parameters on the RECEIVE command, to test your exit routines and debug user-written control records. (See *z/OS TSO/E Customization* for detailed information on the exit routines and control records.)

The syntax of the TRANSMIT command is:

```
          ┌ ┌OUTDDNAME(ddname)┐ ┐
          │ └OUTFILE(ddname)  ┘ │
TRANSMIT <                       >
          │ ┌OUTDSNAME(dsn)   ┐ │
          └ └OUTDATASET(dsn)  ┘ ┘
```

- TRANSMIT is an authorized command.

OUTDDNAME | OUTFILE
A pre-allocated file used as the output data set for the TRANSMIT command. TSO/E does not write any data to SYSOUT. The data set has the following DCB attributes: LRECL=80, BLKSIZE=3120, RECFM=FB.

ddname
The name on a DD statement that identifies the data set. The name may identify a sequential data set or a member of a partitioned data set.

`OUTDSNAME | OUTDATASET`
A data set used as the output data set for the TRANSMIT command. TSO/E does not write any data to SYSOUT. The data set has the following DCB attributes: LRECL=80, BLKSIZE=3120, RECFM=FB.

`dsn`
The name of a **sequential** data set.

value: a name that conforms to TSO/E data set naming conventions

UADSREFM command

The complete description and function of the UADSREFM command are described in *z/OS TSO/E Customization*.

UADSREFM command return codes

Table 8. UADSREFM command return codes

0	Processing successful.
12	Processing unsuccessful. An error message has been issued.

VLFNOTE command

The syntax and function of the VLFNOTE keywords that do not require OPERATOR authority to use are described in *z/OS TSO/E Command Reference*. The keywords described here are intended for your use. You may use them to notify the virtual lookaside facility (VLF) to delete (remove from use through VLF):
- an entire class of IBM-supplied or user-supplied data
- a named collection of data (non-PDS data) from an IBM-supplied class
- partitioned data sets (PDSs) associated with a particular volume serial from an IBM-supplied or user-supplied class

A **class** may be:
- a set of related PDSs - as specified with the EDSN keyword in the COFVLFxx parmlib member
- a named collection of data (non-PDS data) - as specified with the EMAJ keyword in the COFVLFxx parmlib member

An IBM-supplied class name begins with an alphabetic character in the range A-I.

For detailed information concerning the COFVLFxx member of SYS1.PARMLIB, see *z/OS MVS Initialization and Tuning Reference*.

Note: VLF can run on a sysplex and be automatically notified when partitioned data sets are deleted. In this environment, you might not need to enter the VLFNOTE command. (However, VLF is not automatically notified when non-PDS data is deleted.) *z/OS MVS Programming: Authorized Assembler Services Guide* provides more information about VLF notification.

The syntax of the VLFNOTE command is:

```
                 ┌ CLASS(class-name)  [ MAJOR(major-name) ] ┐
VLFNOTE  DELETE ─┤                                          ├
                 └ VOLSER(vol-id)                           ┘
```

- To use the VLFNOTE command, VLF must be installed and active.
- VLFNOTE is an authorized command.
- You require OPERATOR authority to issue the indicated DELETE requests.

DELETE
Delete an entire class of data, non-PDS data from an IBM-supplied class, or all PDSs associated with a particular volume serial.

CLASS
A class of data sets or a class of non-PDS data

class_name
The name of the class

value: a *class-name* specified in the COFVLFxx parmlib member

MAJOR
Non-PDS data from an IBM-supplied class

major_name
The name of the data

value: a *major_name* specified on the EMAJ keyword with the corresponding *class_name* in the COFVLFxx parmlib member

VOLSER
All PDSs associated with the particular volume serial

vol_id
The volume serial of the volume on which the PDSs reside

value: a volume serial known to VLF

Example 1

Operation: Notify VLF to delete data from an IBM-supplied class.

```
vlfnote delete class(amacros) major(ihaqcb)
```

Example 2

Operation: Notify VLF to delete an entire IBM-supplied class.

```
vlfnote delete class(htsomac)
```

Example 3

Operation: Notify VLF to delete an entire user-specified class.

```
vlfnote delete class(user5)
```

Example 4

Operation: Notify VLF to delete all PDSs on the volume.

```
vlfnote delete volser(87-pay)
```

Chapter 3. Information Center Facility trace commands

You can use the trace commands provided with the Information Center Facility to diagnose problems with the facility's CLISTs and REXX execs.

Control tracing by entering one of the following four commands on the OPTION line on any of the Information Center Facility selection panels:
- TRACE1
- TRACE2
- TRACE3.*membername* (where *membername* is the name of a nested CLIST or REXX exec)
- TRACEOFF

Note: The ICQCLM00 and ICQABM30 panels are *not* selection panels; therefore, do not enter trace commands on their OPTION lines.

TRACE1

Use the TRACE1 command to trace the control flow between (or among) nested CLISTs/REXX execs and to display the order of CLIST/REXX exec invocation and execution.

Enter TRACE1 on the OPTION line of the selection panel.

TSO/E redisplays the particular panel with the command still on the OPTION line and with the following message:

```
CLIST or REXX exec tracing will be at level 1
```

TRACE2

Use the TRACE2 command to perform the same functions as TRACE1 and to also display:
- **for all nested CLISTs:** each CLIST statement and TSO/E command and subcommand before execution
- **for all nested REXX execs:** each clause before execution

Enter TRACE2 on the OPTION line of the selection panel.

TSO/E redisplays the particular panel with the command still on the OPTION line and with the following message:

```
CLIST or REXX exec tracing will be at level 2
```

TRACE3.membername

Use the TRACE3 command to perform the same functions as TRACE1 and to also display:
- **for a single explicitly named CLIST:** each CLIST statement and TSO/E command and subcommand before execution

- **for a single explicitly named REXX exec:** each clause before execution

Enter TRACE3.*membername* on the OPTION line of the selection panel.

membername
The name of a nested CLIST or REXX exec

TSO/E redisplays the particular panel with the command still on the OPTION line and with the following message:

```
CLIST/REXX exec member membername will be traced on the screen.
```

If you enter a TRACE3 command without specifying *membername*, TSO/E turns tracing off and displays the following message with the alarm option:

```
CLIST or REXX exec member name required.  TRACEOFF is set.
```

TRACEOFF

Use the TRACEOFF command to deactivate tracing.

Enter TRACEOFF on the OPTION line of the selection panel.

TSO/E redisplays the particular panel with the command still on the OPTION line and with the following message:

```
CLIST or REXX exec trace is turned off.
```

If you enter TRACEOFF when trace is not active, TSO/E displays the following message with the alarm option:

```
CLIST or REXX exec trace is not active.
```

Appendix. Accessibility

Accessible publications for this product are offered through the z/OS Information Center, which is available at www.ibm.com/systems/z/os/zos/bkserv/.

If you experience difficulty with the accessibility of any z/OS information, please send a detailed message to mhvrcfs@us.ibm.com or to the following mailing address:

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Accessibility features

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. Refer to *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.

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the z/OS Information 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 refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? 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.
- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates 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 only applies 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.

- * 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.

  Note:

  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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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.

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: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

Index

Product Number: 5650-ZOS

Printed in USA

SA32-0974-00