

IBM Developer for z Systems
Version 14.1

Host Configuration Reference



Note

Before using this information, be sure to read the general information under [“Notices”](#) on page 35.

Fourth edition (March 2019)

This edition applies to IBM® Developer for z Systems® Version 14.1 and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send your comments by mail to the following address:

IBM Corporation
Attn: Information Development Department 53NA
Building 501 P.O. Box 12195
Research Triangle Park NC 27709-2195
USA

You can fax your comments to: 1-800-227-5088 (US and Canada)

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

Note to U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© **Copyright International Business Machines Corporation 2015, 2019.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures.....	V
Tables.....	vii
About this document.....	ix
Who should use this document.....	ix
Part 1. Host Configuration Reference.....	1
Chapter 1. Understanding Developer for z Systems.....	3
Component overview.....	3
Task owners.....	4
CARMA.....	5
CARMA configuration files.....	6
z/OS UNIX directory structure.....	7
Chapter 2. Security considerations.....	9
Authentication methods.....	9
Connection security.....	9
SCLM security.....	9
Security definitions.....	10
Requirements and checklist.....	10
Define the data set profiles.....	10
Verify the security settings.....	11
Chapter 3. TCP/IP considerations.....	13
TCP/IP ports.....	13
External communication.....	13
Internal communication.....	14
TCP/IP port reservation.....	14
CARMA and TCP/IP	14
CARMA and TCP/IP ports.....	14
CARMA and stack affinity.....	15
Chapter 4. WLM considerations.....	17
Workload classification.....	17
Classification rules.....	18
Setting goals.....	19
Considerations for goal selection.....	19
OMVS.....	20
JES.....	21
Chapter 5. Push-to-client considerations.....	23
Introduction.....	23
Host-based projects.....	23
Chapter 6. CICSTS considerations.....	25
xUnit support for CICS applications.....	25
Bidirectional language support.....	25

Diagnostic IRZ messages for Enterprise Service Tools.....	25
Chapter 7. SMF considerations.....	27
SMF type 122, subtype 1.....	27
Client activation code request.....	27
Header section.....	27
Data section 1, Creator ID.....	30
Data section 2, Server initialization.....	30
Data section 3, VU license handler status.....	31
Data section 4, Client UUID.....	32
Data section 5, Client labels.....	32
Data section 6, Client data.....	33
Notices.....	35
Programming interface information.....	36
Trademarks.....	36
Terms and conditions for product documentation.....	36

Figures

- 1. Component overview.....3
- 2. Task owners..... 4
- 3. CARMA flow..... 5
- 4. z/OS UNIX directory structure..... 7
- 5. TCP/IP ports..... 13
- 6. WLM classification.....17

Tables

- 1. Security setup variables..... 10
- 2. WLM entry-point subsystems..... 18
- 3. WLM work qualifiers..... 18
- 4. WLM workloads..... 19
- 5. WLM workloads - OMVS..... 20
- 6. WLM workloads - JES..... 21
- 7. SMF record type 122 subtype 1, header section..... 28
- 8. SMF record type 122 subtype 1, data section 1.....30
- 9. SMF record type 122 subtype 1, data section 2.....31
- 10. SMF record type 122 subtype 1, data section 3..... 31
- 11. SMF record type 122 subtype 1, data section 4..... 32
- 12. SMF record type 122 subtype 1, data section 5..... 32
- 13. SMF record type 122 subtype 1, data section 6..... 33

About this document

This document gives background information for various configuration tasks of IBM Developer for z Systems itself and other z/OS® components and products (such as WLM and TCP/IP).

The following names are used in this manual:

- *IBM Explorer for z/OS* is called *z/OS Explorer*.
- *IBM z/OS Debugger* is called *z/OS Debugger*.
- *IBM Developer for z Systems* is called *Developer for z Systems*.
- *Common Access Repository Manager* is abbreviated to *CARMA*.
- *Software Configuration and Library Manager Developer Toolkit* is called *SCLM Developer Toolkit*, abbreviated to *SCLMDT*.
- *z/OS UNIX System Services* is called *z/OS UNIX*.
- *Customer Information Control System Transaction Server* is called *CICSTS*, abbreviated to *CICS*®.
- *z/OS Automated Unit Testing Framework* is called *zUnit*.

This document is part of a set of documents that describe Developer for z Systems host configuration. Each of these documents has a specific target audience. You are not required to read all documents to complete the Developer for z Systems configuration.

- *IBM Developer for z Systems Host Configuration Guide* describes in detail all planning tasks, configuration tasks and options (including optional ones) and provides alternative scenarios.
- *IBM Developer for z Systems Host Configuration Reference* describes Developer for z Systems design and gives background information for various configuration tasks of Developer for z Systems, z/OS components, and other products (such as WLM and TCP/IP) related to Developer for z Systems.

For the most up-to-date versions of this document, see the Developer for z Systems Knowledge Center available at https://www.ibm.com/support/knowledgecenter/SSQ2R2/rdz_welcome.html.

For the most up-to-date versions of the complete documentation, including installation instructions, white papers, podcasts, and tutorials, see the [library page of the IBM Developer for z Systems website](http://www.ibm.com/support/docview.wss?uid=swg27048563) (<http://www.ibm.com/support/docview.wss?uid=swg27048563>).

Who should use this document

This document is intended for system programmers configuring and tuning IBM Developer for z Systems.

While the actual configuration steps are described in the *Host Configuration Guide*. This publication lists in detail various related subjects, such as tuning, security setup, and more. To use this document, you must be familiar with the z/OS UNIX System Services and MVS™ host systems.

Part 1. Host Configuration Reference

Chapter 1. Understanding Developer for z Systems

The Developer for z Systems host consists of several components that interact to give the client access to the host services and data. Understanding the design of these components can help you make the correct configuration decisions.

The following topics are covered in this chapter:

- [“Component overview” on page 3](#)
- [“Task owners” on page 4](#)
- [“CARMA” on page 5](#)
- [“z/OS UNIX directory structure” on page 7](#)

Developer for z Systems builds on top of IBM Explorer for z/OS. For z/OS Explorer related information, see "Understanding z/OS Explorer" in the *IBM Explorer for z/OS Host Configuration Reference*.

Component overview

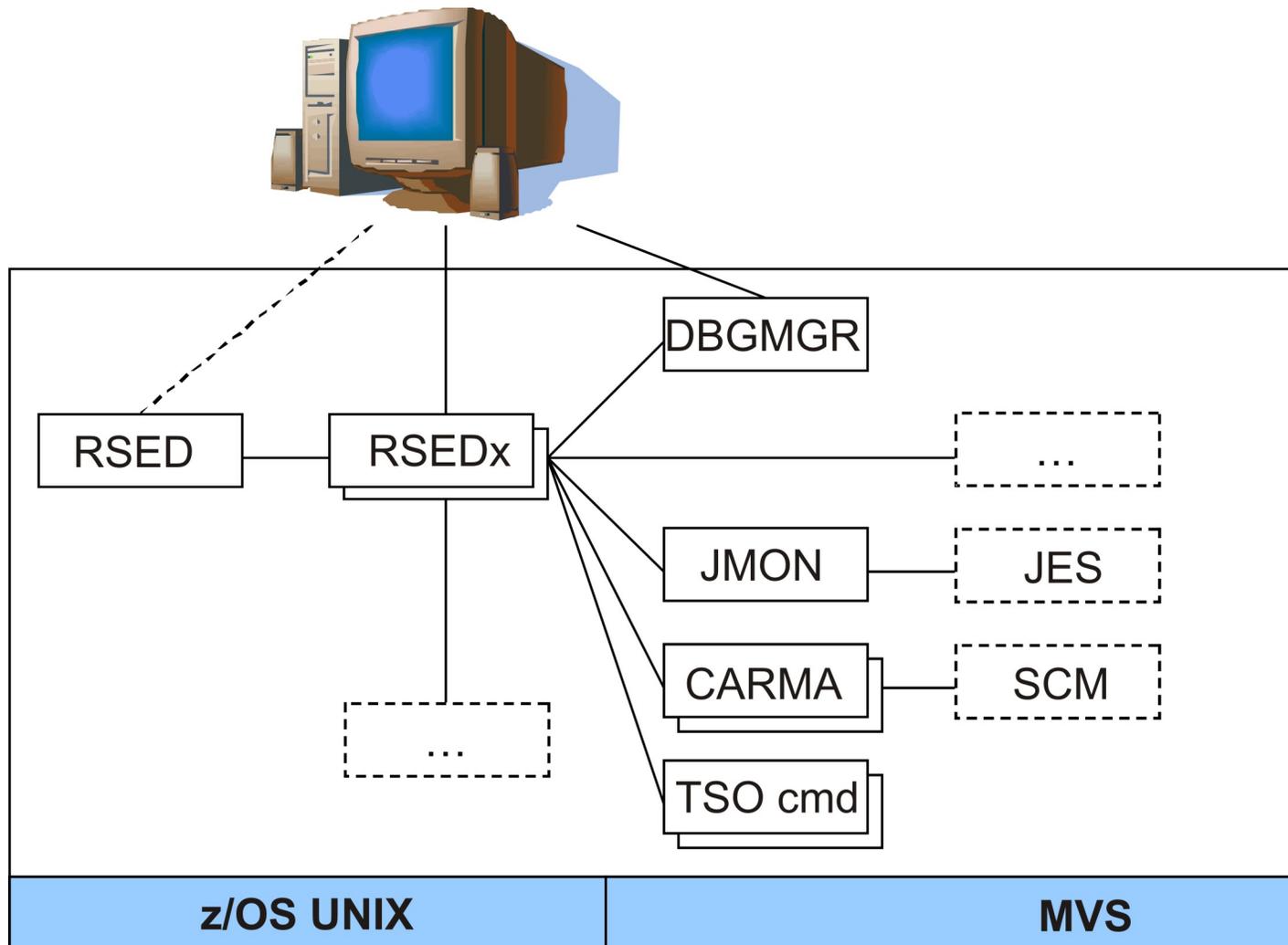


Figure 1. Component overview

Figure 1 on page 3 shows a generalized overview of the combined z/OS Explorer, z/OS Debugger, and Developer for z Systems layout on your host system.

- (z/OS Explorer) Remote Systems Explorer (RSE) provides core services, such as connecting the client to the host and starting other servers for specific services. RSE consists of two logical entities:
 - RSE daemon (RSED), which manages connection setup. RSE daemon is also responsible for running in single server mode. To do so, RSE daemon creates one or more child processes known as RSE thread pools (RSEDx).
 - RSE server, which handles individual client request. An RSE server is active as a thread inside a RSE thread pool.
- (z/OS Debugger) Debug Manager (DBGMGR) coordinates debugger activity.
- (z/OS Explorer) TSO Commands Service (TSO cmd) provides a batch-like interface for TSO and ISPF commands.
- (z/OS Explorer) JES Job Monitor (JMON) provides all JES related services.
- Common Access Repository Manager (CARMA) provides an interface to interact with Software Configuration Managers (SCMs), such as CA Endeavor® SCM.
- More services are available, which can be provided by Developer for z Systems itself or corequisite software.

The description in the previous paragraph and list shows the central role assigned to RSE. With few exceptions, all client communication goes through RSE. This allows for easy security related network setup, as only a limited set of ports are used for client-host communication.

To manage the connections and workloads from the clients, RSE is composed of a daemon address space, which controls thread pool address spaces. The daemon acts as a focal point for connection and management purposes, while the thread pools process the client workloads. Based upon the values defined in the `rse.env` configuration file, and the amount of actual client connections, multiple thread pool address spaces can be started by the daemon.

Task owners

Figure 2 on page 4 shows a basic overview of the owner of the security credentials used for various z/OS Explorer, z/OS Debugger, and Developer for z Systems tasks.

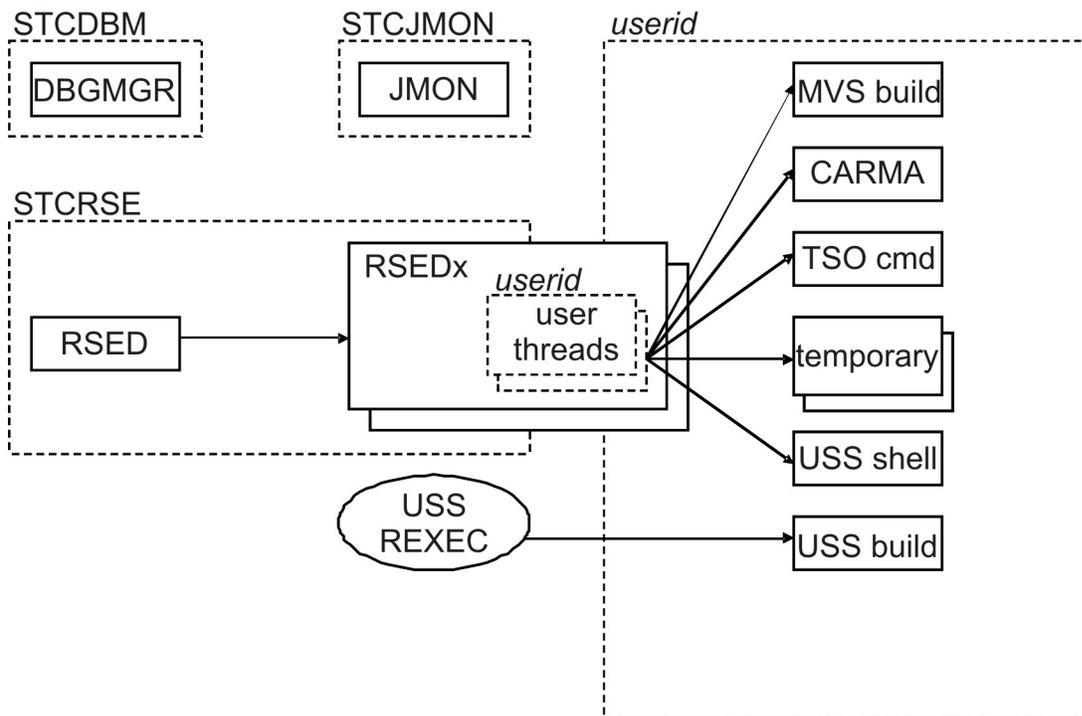


Figure 2. Task owners

The ownership of a task can be divided into two sections. Started tasks are owned by the user ID that is assigned to the started task in your security software. All other tasks, with the RSE thread pools (RSEDx) as exception, are owned by the client user ID.

Figure 2 on page 4 shows the z/OS Explorer, z/OS Debugger, and Developer for z Systems started tasks (DBGMGR, JMON, and RSED), and sample started tasks and system services that Developer for z Systems communicates with. The USS REXEC tag represents the z/OS UNIX REXEC (or SSH) service.

RSE daemon (RSED) creates one or more RSE thread pool address spaces (RSEDx) to process client requests. Each RSE thread pool supports multiple clients and is owned by the same user as the RSE daemon. Each client has his own threads inside a thread pool, and these threads are owned by the client user ID.

Depending on actions done by the client, one or more additional address spaces can be started, all owned by the client user ID, to perform the requested action. These address spaces can be an MVS batch job, or a z/OS UNIX child process. Note that a z/OS UNIX child process is active in a z/OS UNIX initiator (BPXAS), and the child process shows up as a started task in JES.

The creation of these address spaces is most often triggered by a user thread in a thread pool, either directly or by using system services like ISPF. But the address space could also be created by a third party. For example, z/OS UNIX REXEC or SSH are involved when starting builds in z/OS UNIX.

The user-specific address spaces end at task completion or when an inactivity timer expires. The started tasks remain active. The address spaces listed in Figure 2 on page 4 remain in the system long enough to be visible. However, you should be aware that due to the way z/OS UNIX is designed, there are also several short-lived temporary address spaces.

CARMA

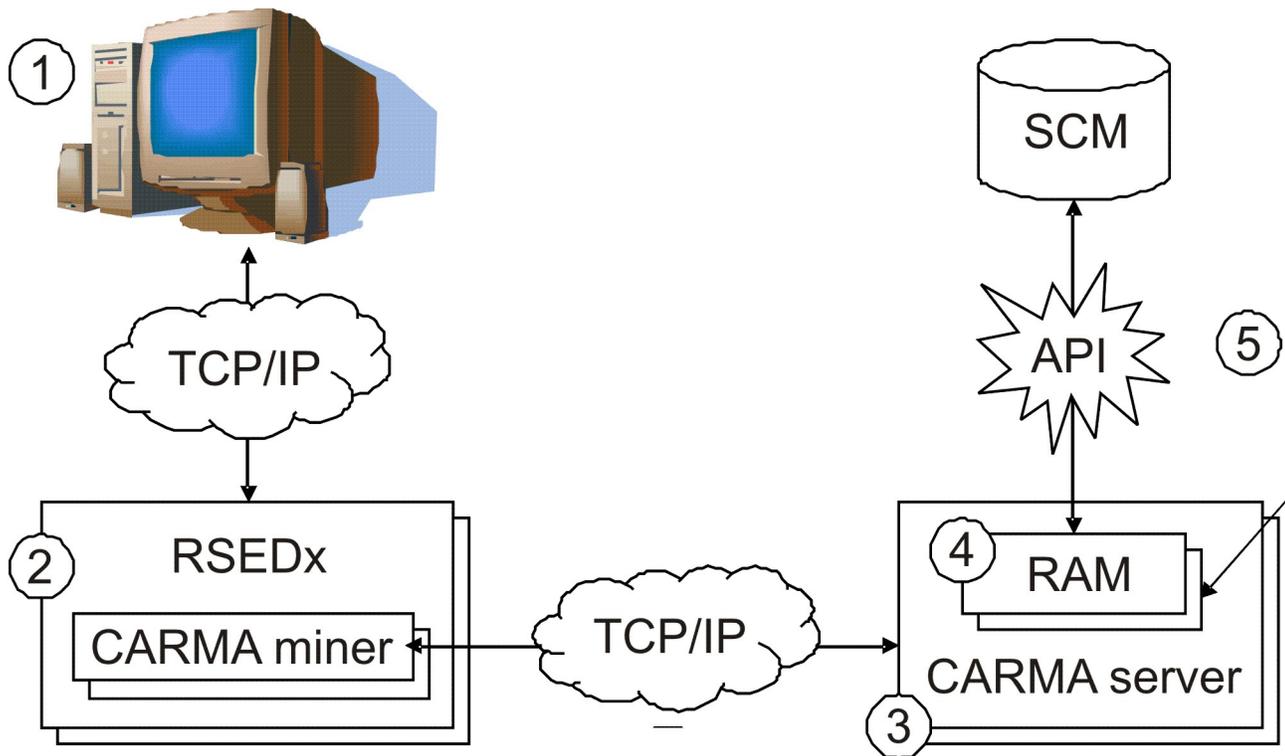


Figure 3. CARMA flow

CARMA (Common Access Repository Manager) is used to access a host based Software Configuration Manager (SCM), for example CA Endevor® SCM. Figure 3 on page 5 shows a schematic overview of how a

Developer for z Systems client can access any supported host-based Software Configuration Manager (SCM).

1. The client has a Common Access Repository Manager (CARMA) plugin.
2. The CARMA plugin communicates with the CARMA miner, active as a user-specific thread within the RSE thread pool (RSEdX). This communication is done by way of the existing RSE connection.
3. When the client requests access to a SCM, the CARMA miner will bind to a TCP/IP port and will start a user-specific CARMA server, with the port number as startup argument. The CARMA server will then connect to this port and use this path for communication with the client. Note that host based SCMs expect single-user address spaces to access their services, which requires CARMA to start a CARMA server per user. It is not possible to create a single server supporting multiple users.
4. The CARMA server will load the Repository Access Manager (RAM) that supports the requested SCM.
5. The RAM deals with the technical details of interacting with the specific SCM, and presents a common interface to the client.

CARMA configuration files

Developer for z Systems supports multiple methods to start a CARMA server. Each method has benefits and drawbacks. Developer for z Systems also provides multiple Repository Access Managers (RAMs), which can be divided into two groups, production RAMs and sample RAMs. Various combinations of RAMs and server startup methods are available as a preconfigured setup.

All server startup methods share a common configuration file, `CRASRV.properties`, which (among other things) specifies which startup method will be used.

CRASTART

The "CRASTART" method starts the CARMA server as a subtask within RSE. It provides a very flexible setup by using a separate configuration file that defines data set allocations and program invocations needed to start a CARMA server. This method provides the best performance and uses the fewest resources, but requires that module CRASTART is located in LPA.

RSE invokes load module CRASTART, which uses the definitions in `crastart*.conf` to create a valid environment to execute batch TSO and ISPF commands. Developer for z Systems uses this environment to run the CARMA server, CRASERV. Developer for z Systems provides multiple `crastart*.conf` files, each preconfigured for a specific RAM.

Batch submit

The "batch submit" method starts the CARMA server by submitting a job. This is the default method used in the provided sample configuration files. The benefit of this method is that the CARMA logs are easily accessible in the job output. It also allows the use of custom server JCL for each developer, which is maintained by the developer himself. However, this method uses one JES initiator per developer starting a CARMA server.

RSE invokes CLIST CRASUB*, which in turn submits an embedded JCL to create a valid environment to execute batch TSO and ISPF commands. Developer for z Systems uses this environment to run the CARMA server, CRASERV. Developer for z Systems provides multiple CRASUB* members, each preconfigured for a specific RAM.

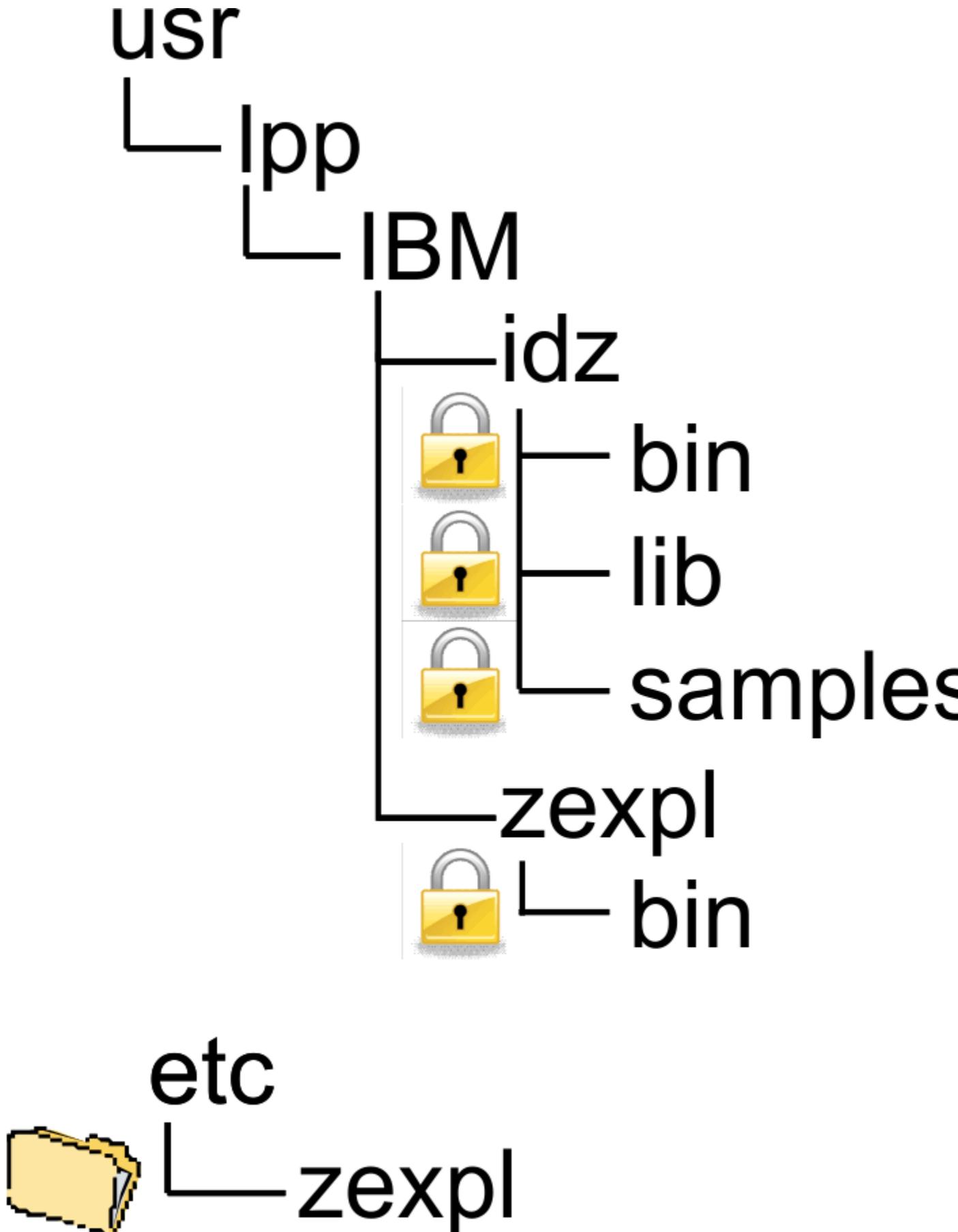


Figure 4 on page 7 shows an overview of the z/OS UNIX directories used by Developer for z Systems. The following list describes each directory touched by Developer for z Systems, how the location can be changed, and who maintains the data within.

- `/usr/lpp/IBM/idz/` is the root path for the Developer for z Systems product code. The actual location is specified in the `idz.env` configuration file (variable `IDZ_HOME`). The files within are maintained by SMP/E.
- Developer for z Systems places files in `/usr/lpp/IBM/zexpl/bin`, the binaries directory of z/OS Explorer. The actual location is specified in the z/OS Explorer configuration. The files within are maintained by SMP/E.
- `/etc/zexpl/` holds the z/OS Explorer and Developer for z Systems configuration files. The actual location is specified in the RSED started task (variable `CNFG`). The files within are maintained by the system programmer.
- `/tmp/` is used by Legacy ISPF Gateway to store temporary data. Some IVPs store their output here. The files within are maintained by ISPF and the IVPs. The location can be customized with the `TMPDIR` variable in `rse.env`. It is also the default location for Java™ dump files, which can be customized with the `_CEE_DUMPTARG` variable in `rse.env`.

Note: `/tmp/` requires permission bit mask 777 to allow each client to create temporary files.

- `/var/zexpl/WORKAREA/` is used by Legacy ISPF Gateway and SCLMDT to transfer data between z/OS UNIX and MVS based address spaces. The actual location is specified in `rse.env` (variable `CGI_ISPWORK`). The files within are maintained by ISPF and SCLMDT.

Note: `/var/zexpl/WORKAREA/` requires permission bit mask 777 to allow each client to create temporary files.

- Developer for z Systems writes log messages in the z/OS Explorer log files located in `/var/zexpl/zexpl/logs/$LOGNAME`. The actual location is specified in the z/OS Explorer configuration. The files within are maintained by z/OS Explorer and Developer for z Systems product code.
- The host components of Developer for z Systems are shared by multiple products, and the data in `/var/idz/sclmdt` is not used by Developer for z Systems.
- `/var/idz/sclmdt/CONFIG/` holds general SCLMDT configuration files. The actual location is specified in `idz.env` (variable `SCLMDT_CONF_HOME`). The files within are maintained by the SCLM administrator.
- `/var/idz/sclmdt/CONFIG/PROJECT/` holds SCLMDT project configuration files. The actual location is specified in `idz.env` (variable `SCLMDT_CONF_HOME`). The files within are maintained by the SCLM administrator.
- `/var/idz/sclmdt/CONFIG/script/` holds SCLMDT-related scripts that can be used by other products. The actual location is not specified anywhere. The files within are maintained by the SCLM administrator.
- `/var/zexpl/pushtoclient/` holds host-based project information, client configuration files and client product update information that is pushed to the client upon connection to the host. The actual location is specified in `pushtoclient.properties` (variable `pushtoclient.folder`). The files within are maintained by a Developer for z Systems client administrator.

Note: The host components of Developer for z Systems are shared by multiple products, and the data in `/var/zexpl/pushtoclient/projects` is not used by Developer for z Systems.

- `/var/zexpl/pushtoclient/projects/` holds the host-based project definition files. The actual location is specified in `/var/zexpl/pushtoclient/keymapping.xml`, which is created and maintained by a Developer for z Systems client administrator. The files within are maintained by a project manager or lead developer.

Chapter 2. Security considerations

Developer for z Systems extends z/OS Explorer by providing additional functions, some of which interact with other system components and products, such as CICS. Developer for z Systems specific security definitions are used to secure the provided functions.

The security mechanisms used by Developer for z Systems servers and services rely on the data sets and file systems it resides in being secure. This implies that only trusted system administrators should be able to update the program libraries and configuration files.

Developer for z Systems builds on top of IBM Explorer for z/OS. For z/OS Explorer related information, see "Security consideration" in the *IBM Explorer for z/OS Host Configuration Reference*.

The following topics are covered in this chapter:

- [“Authentication methods” on page 9](#)
- [“Connection security” on page 9](#)
- [“SCLM security” on page 9](#)
- [“Security definitions”](#)

Authentication methods

CARMA authentication

Client authentication is done by RSE daemon as part of the client's connection request. CARMA is started from a user specific thread, and inherits the user's security environment, bypassing the need for additional authentication.

SCLM Developer Toolkit authentication

Client authentication is done by RSE daemon as part of the client's connection request. SCLMDT is started from a user specific thread, and inherits the user's security environment, bypassing the need for additional authentication.

Connection security

Most communication between Developer for z Systems client and host goes through RSE, thus utilizing the connection security provided by z/OS Explorer.

Some Developer for z Systems services use a separate external (client-host) communication path:

- The Host Connect Emulator on the client connects to a TN3270 server on the host. The encryption details are controlled by an Application Transparent Transport Layer Security (AT-TLS) policy.
- Remote (host-based) actions in z/OS UNIX subprojects use an REXEC or SSH server on the host. SSH communication is always encrypted.

SCLM security

The SCLM Developer Toolkit service offers optional security functionality for the Build, Promote, and Deploy functions.

If security is enabled for a function by the SCLM administrator, SAF calls are made to verify authority to execute the protected function with the caller's or a surrogate user ID.

Refer to *SCLM Developer Toolkit Administrator's Guide* (SC23-9801), for more information on the required SCLM security definitions.

Security definitions

Customize and submit the sample FELRACF job, which has sample RACF® commands to create the basic security definitions for Developer for z Systems.

FELRACF is located in FEL.#CUST.JCL, unless you specified a different location when you customized and submitted the FEL.SFELSAMP (FELSETUP) job. See "Customization setup" in the *Host Configuration Guide* for more details.

See the *RACF Command Language Reference (SA22-7687)* for more information about RACF commands.

Requirements and checklist

To complete the security setup, the security administrator must know the values that are listed in [Table 1](#) on page 10. These values were defined during previous steps of the installation and customization of Developer for z Systems.

Description	<ul style="list-style-type: none">• Default value• Where to find the answer	Value
Developer for z Systems product high-level qualifier	<ul style="list-style-type: none">• FEL• SMP/E installation	
Developer for z Systems customization high-level qualifier	<ul style="list-style-type: none">• FEL.#CUST• FEL.SFELSAMP (FELSETUP), as described in "Customization setup" in the <i>Host Configuration Guide</i>.	

The following list is an overview of the actions that are required to complete the basic security setup of Developer for z Systems. As documented in the following sections, different methods can be used to fulfill these requirements, depending on the required security level.

- ["Define the data set profiles" on page 10](#)
- ["Verify the security settings" on page 11](#)

Define the data set profiles

READ access for users and ALTER for system programmers is sufficient for most Developer for z Systems data sets. Replace the #sysprog placeholder with valid user IDs or RACF group names. Also, ask the system programmer who installed and configured the product for the correct data set names. FEL is the default high-level qualifier used during installation and FEL.#CUST is the default high-level qualifier for data sets created during the customization process.

- ```
ADDGROUP (FEL) OWNER(IBMUSER) SUPGROUP(SYS1)
DATA('IBM Developer for z Systems - HLQ STUB')
```
- ```
ADDSD 'FEL.*.**' UACC(READ)
DATA('IBM Developer for z Systems')
```
- ```
PERMIT 'FEL.*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
```
- ```
SETROPTS GENERIC(DATASET) REFRESH
```

Notes:

- Protect FEL . SFELLPA against updates because this data set is loaded into LPA, which is APF authorized by default.
- The sample commands in this publication and in the FELRACF job assume that Enhanced Generic Naming (EGN) is active. When EGN is active, the ** qualifier can be used to represent any number of qualifiers in the DATASET class. Substitute ** with * if EGN is not active on your system. For more information about EGN, see *Security Server RACF Security Administrator's Guide (SA22-7683)*.

Some of the Developer for z Systems components require additional security data set profiles. Replace the #sysprog and #ram-developer placeholders with valid user ID's or RACF group names:

- If SCLM Developer Toolkit's long/short name translation is used, users require UPDATE access to the mapping VSAM, FEL . #CUST . LSTRANS . FILE.

```
- ADDSD 'FEL.#CUST.LSTRANS.*.**' UACC(UPDATE)
  DATA('IBM Developer for z Systems - SCLMDT')
```

```
- PERMIT 'FEL.#CUST.LSTRANS.*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
```

```
- SETROPTS GENERIC(DATASET) REFRESH
```

- CARMA RAM (Repository Access Manager) developers require UPDATE access to the CARMA VSAMs, FEL . #CUST . CRA*.

```
- ADDSD 'FEL.#CUST.CRA*.**' UACC(READ)
  DATA('IBM Developer for z Systems - CARMA')
```

```
- PERMIT 'FEL.#CUST.CRA*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
```

```
- PERMIT 'FEL.#CUST.CRA*.**' CLASS(DATASET) ACCESS(UPDATE) ID(#ram-developer)
```

```
- SETROPTS GENERIC(DATASET) REFRESH
```

Verify the security settings

Use the following sample commands to display the results of your security-related customizations.

- Data set profiles
 - LISTGRP FEL
 - LISTDSD PREFIX(FEL) ALL

Chapter 3. TCP/IP considerations

Developer for z Systems uses TCP/IP to provide mainframe access to users on a non-mainframe workstation. It also uses TCP/IP for communication between various components and other products.

The following topics are covered in this chapter:

- “TCP/IP ports” on page 13
- “CARMA and TCP/IP ports” on page 14

Developer for z Systems builds on top of IBM Explorer for z/OS. For z/OS Explorer related information, see “TCP/IP considerations” in the *IBM Explorer for z/OS Host Configuration Reference*.

TCP/IP ports

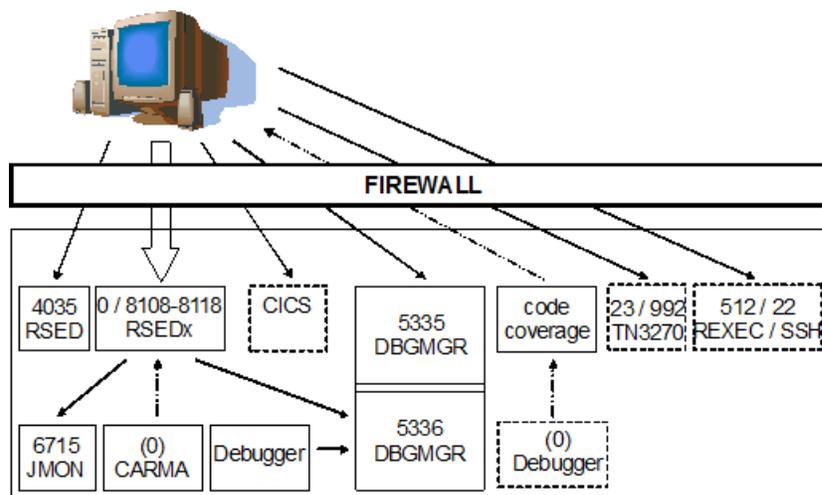


Figure 5. TCP/IP ports

Figure 5 on page 13 shows the TCP/IP ports that can be used by z/OS Explorer, z/OS Debugger, and Developer for z Systems. The arrows show which party does the bind (arrowhead side) and which one connects.

External communication

Define the following ports to your firewall protecting the z/OS host, as they are used for client-host communication (using the tcp protocol):

- (z/OS Explorer) RSE daemon for client-host communication setup, default port 4035. The port can be set in the `rse.env` configuration file. Communication on this port can be encrypted.
- (z/OS Explorer) RSE server for client-host communication. By default, any available port is used, but this can be limited to a specified range with the `_RSE_PORTRANGE` definition in `rse.env`. The default port range for `_RSE_PORTRANGE` is 8108-8118 (11 ports). Communication on this port can be encrypted.
- (z/OS Debugger) Debug manager for debugger services, default port 5335. The port can be set in the DBGMGR started task JCL. Communication on this port can be encrypted.
- CICS PIPELINE port for zUnit CICS recorder. There is no default. The port can be set in the CICS CSD. Communication on this port can be encrypted.
- Either INETD service for remote (host-based) actions in z/OS UNIX subprojects:
 - REXEC (z/OS UNIX version), default port 512.
 - SSH (z/OS UNIX version), default port 22. Communication on this port is encrypted.

- TN3270 Telnet service for the Host Connect Emulator, default port 23. Communication can be encrypted (default port 992). The default port assigned to the TN3270 Telnet service depends on whether or not the user chooses to use encryption.
- Host-based code coverage can be instructed to connect to the Engine of a Developer for z Systems client. Communication on this port can be encrypted. Note that in this scenario, the z/OS-based code coverage collector is a client for TCP/IP and the Engine on the user's personal computer is a server for TCP/IP. The default is to work with z/OS Debugger locally on the same host.

Note: Normally the client specifies which TCP/IP address is used to connect to the host. However, to ensure that debug sessions communicate with the correct host, the Debug Manager dictates to the client which TCP/IP address must be used.

Internal communication

Several Developer for z Systems host services run in separate threads or address spaces and are using TCP/IP sockets as communication mechanism, using your system's loopback address. All these services use RSE for communicating with the client, making their data stream confined to the host only. For some services any available port will be used, for others the system programmer can choose the port or port range that will be used:

- (z/OS Explorer) JES Job Monitor for JES-related services, default port 6715. The port can be set in the FEJCCNFG configuration member and is repeated in the `rse.env` configuration file.
- CARMA communication uses by default an ephemeral port, but a port range can be set in the `CRASRV.properties` configuration file.
- (z/OS Debugger) Debug Manager for debug related services, default port 5336. The port can be set in the DBGMGR started task JCL.
- Host-based code coverage, which is a batch job, allocates an ephemeral port to allow the z/OS Debugger to communicate with it and deliver data needed for the code coverage report.

TCP/IP port reservation

If you use the `PORT` or `PORTRANGE` statement in `PROFILE.TCPIP` to reserve the ports used by z/OS Explorer, z/OS Debugger, and Developer for z Systems, note that many binds are done by threads active in an RSE thread pool. The job name of the RSE thread pool is `RSEdx`, where `RSEd` is the name of the RSE started task, and `x` is a random single digit number, so wildcards are required in the definition.

```
PORT      4035      TCP RSED   ; z/OS Explorer - RSE daemon
PORT      6715      TCP JMON   ; z/OS Explorer - JES job monitor
PORT      5335      TCP DBGMGR ; z/OS Debugger - debug manager
PORT      5336      TCP DBGMGR ; z/OS Debugger - debug manager
PORTRange 8108 11  TCP RSED*  ; z/OS Explorer - RSE_PORTRANGE
;PORTRange 5227 100 TCP RSED*  ; Developer for z Systems - CARMA
```

CARMA and TCP/IP

CARMA and TCP/IP ports

CARMA (Common Access Repository Manager) is used to access a host-based Software Configuration Manager (SCM), for example CA Endeavor® SCM. In most cases, like for RSE daemon, a server binds to a port and listens for connection requests. CARMA however uses a different approach, as the CARMA server is not active yet when the client initiates the connection request.

When the client sends a connection request, the CARMA miner, which is active as a user thread in an RSE thread pool, will request an ephemeral port or find a free port in the range specified in the `CRASRV.properties` configuration file and binds to it. The miner then starts the CARMA server and passes the port number, so that the server knows to which port to connect. When the server is connected, the client can send requests to the server and receive the results.

From a TCP/IP perspective, RSE (by way of the CARMA miner) is the server that binds to the port, and the CARMA server is the client connecting to it.

If you use the PORT or PORTRANGE statement in PROFILE.TCPIP to reserve the port range used by CARMA, note that the CARMA miner is active in an RSE thread pool. The jobname of the RSE thread pool is RSEDx, where RSED is the name of the RSE started task and x is a random single digit number, so wildcards are required in the definition.

```
PORTRange 5227 100 RSED*          ; Developer for z
Systems - CARMA
```

Note: The CARMA IVP, `felfivpc`, will fail if you reserve the CARMA ports for usage by the RSE address spaces. This is to be expected because the IVP runs in the address space of the person executing the IVP, not in RSE's address space, and TCP/IP will fail the bind request.

CARMA and stack affinity

CARMA (Common Access Repository Manager) is used to access a host-based Software Configuration Manager (SCM), for example CA Endeavor® SCM. To do so, CARMA starts a user-specific server, which needs additional configuration to enforce stack affinity.

Similar to the z/OS Explorer started tasks, stack affinity for a CARMA server is set with the `_BPXK_SETIBMOPT_TRANSPORT` variable, which must be passed on to LE (Language Environment®). This can be done by adjusting the startup command in the active `crastart*.conf` or `CRASUB*` configuration file.

Note:

- The exact name of the configuration file that holds the startup command depends on various choices made by the systems programmer who configured CARMA. Refer to "Chapter 3. (Optional) Common Access Repository Manager (CARMA)" in the *Host Configuration Guide (SC27-8577)* for more information about this.
- `_BPXK_SETIBMOPT_TRANSPORT` specifies the name of the TCP/IP stack to be used, as defined in the `TCPIPJOBNAME` statement in the related `TCPIP.DATA`.
- Coding a `SYSTCPD DD` statement does not set the requested stack affinity.
- By default, CARMA does not use the normal TCP/IP stacks. CARMA uses the loopback address for the communication between CARMA miner and CARMA server. This improves security (only local processes have access to the loopback address) and is likely to prevent the need to add stack affinity to CARMA communication.

crastart*.conf

Replace the following part:

```
... PARM(&CRAPRM1. &CRAPRM2.)
```

with this (where `TCPIP` represents the desired TCP/IP stack):

```
... PARM(ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIP") / &CRAPRM1. &CRAPRM2.)
```

Note: `CRASTART` does not support line continuations, but there is no limit on the accepted line length.

CRASUB*

Replace the following part:

```
... PARM(&PORT &TIMEOUT)
```

with this (where `TCPIP` represents the desired TCP/IP stack):

```
... PARM(ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIP") / &PORT &TIMEOUT)
```

Note: Job submission limits line length to 80 characters. You can break a longer line at a blank () and use a plus (+) sign at the end of the first line to concatenate 2 lines.

Chapter 4. WLM considerations

Unlike traditional z/OS applications, Developer for z Systems is not a monolithic application that can be identified easily to Workload Manager (WLM). Developer for z Systems consists of several components that interact to give the client access to the host services and data. As described in [Chapter 1, “Understanding Developer for z Systems,”](#) on page 3, some of these services are active in different address spaces, resulting in different WLM classifications.

The following topics are covered in this chapter:

- “[Workload classification](#)” on page 17
- “[Setting goals](#)” on page 19

Developer for z Systems builds on top of IBM Explorer for z/OS. For z/OS Explorer related information, see “WLM considerations” in the *IBM Explorer for z/OS Host Configuration Reference*.

Workload classification

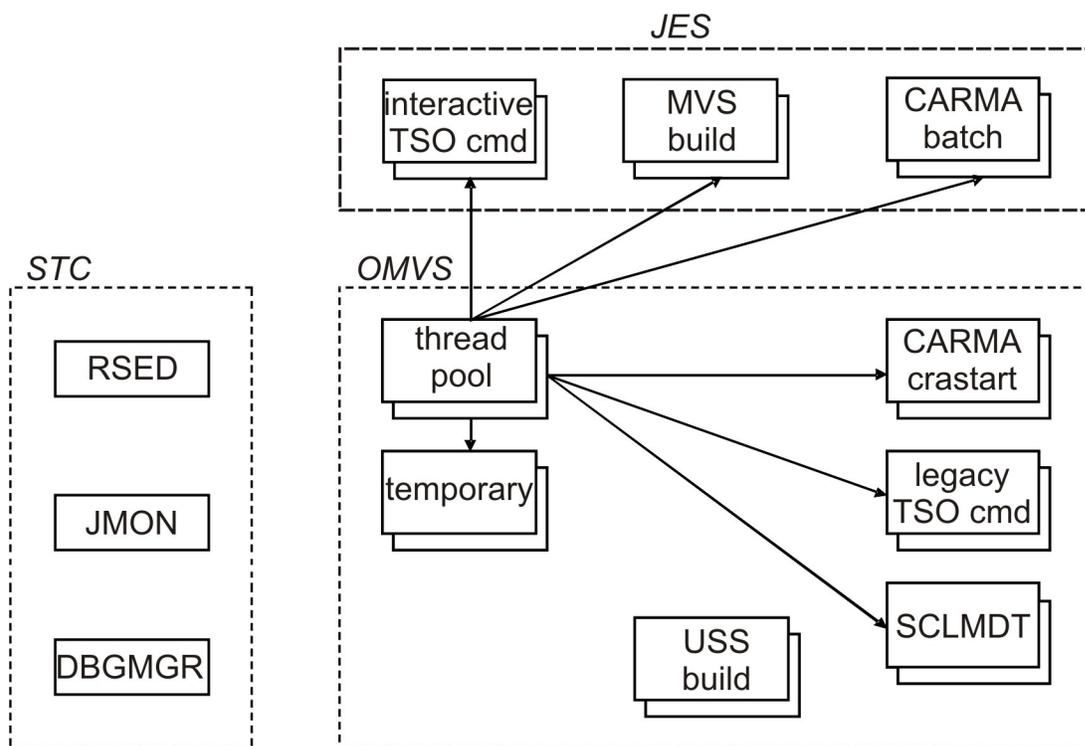


Figure 6. WLM classification

Figure 6 on page 17 shows a basic overview of the subsystems through which z/OS Explorer, z/OS Debugger, and Developer for z Systems workloads are presented to WLM.

RSE daemon (RSED), Debug Manager (DBGMGR) and JES Job Monitor (JMON) are z/OS Explorer and z/OS Debugger started tasks (or long-running batch jobs), each with their individual address space.

RSE daemon spawns a child process for each RSE thread pool server (which supports a variable number of clients). Each thread pool is active in a separate address space (using a z/OS UNIX initiator, BPXAS). Because these are spawned processes, they are classified using the WLM OMVS classification rules, not the started task classification rules.

The clients that are active in a thread pool can create a multitude of other address spaces, depending on the actions done by the users. Depending on the configuration of Developer for z Systems, some

workloads, such as the TSO Commands service (TSO cmd), can run in address spaces with a different classification.

The address spaces listed in Figure 6 on page 17 remain in the system long enough to be visible, but you should be aware that due to the way z/OS UNIX is designed, there are also several short-lived temporary address spaces. These temporary address spaces are active in the OMVS subsystem.

Note that while the RSE thread pools use the same user ID and a similar job name as the RSE daemon, all address spaces started by a thread pool are owned by the user ID of the client requesting the action. The client user ID is also used as (part of) the job name for all OMVS-based address spaces stated by the thread pool.

More address spaces are created by other services that Developer for z Systems uses, such as z/OS UNIX REXEC (USS build).

Classification rules

WLM uses classification rules to map work coming into the system to a service class. This classification is based upon work qualifiers. The first (mandatory) qualifier is the subsystem type that receives the work request. Table 2 on page 18 lists the subsystem types that can receive Developer for z Systems workloads.

Subsystem type	Work description
JES	The work requests include all jobs that JES2 or JES3 initiates.
OMVS	The work requests include work processed in z/OS UNIX System Services forked children address spaces.
STC	The work requests include all work initiated by the START and MOUNT commands. STC also includes system component address spaces.

Table 3 on page 18 lists additional qualifiers that can be used to assign a workload to a specific service class. Refer to MVS Planning: Workload Management (SA22-7602) for more details on the listed work qualifiers.

		JES	OMVS	STC
AI	Accounting Information	x	x	x
LU	LU Name (*)			
PF	Perform (*)	x		x
PRI	Priority	x		
SE	Scheduling Environment Name	x		
SSC	Subsystem Collection Name	x		
SI	Subsystem Instance (*)	x		
SPM	Subsystem Parameter			x
PX	Sysplex Name	x	x	x
SY	System Name (*)		x	x
TC	Transaction/Job Class (*)	x		
TN	Transaction/Job Name (*)	x	x	x
UI	User ID (*)	x	x	x

Note: For the qualifiers marked with (*), you can specify classification groups by adding a G to the type abbreviation. For example, a transaction name group would be TNG.

Setting goals

As documented in “Workload classification” on page 17, Developer for z Systems creates different types of workloads on your system. These different tasks communicate with each other, which implies that the actual elapse time becomes important to avoid time-out issues for the connections between the tasks. As a result, Developer for z Systems tasks should be placed in high-performance service classes, or in moderate-performance service classes with a high priority.

A revision, and possibly an update, of your current WLM goals is therefore advised. This is especially true for traditional MVS shops new to time-critical OMVS workloads.

Note:

- The goal information in this section is deliberately kept at a descriptive level, because actual performance goals are very site-specific.
- To help understand the impact of a specific task on your system, terms like minimal, moderate and substantial resource usage are used. These are all relative to the total resource usage of Developer for z Systems itself, not the whole system.

Table 4 on page 19 lists the address spaces that are used by z/OS Explorer and Developer for z Systems. z/OS UNIX will substitute "x" in the "Task Name" column by a random 1-digit number.

<i>Table 4. WLM workloads</i>		
Description	Task name	Workload
(z/OS Debugger) Debug Manager	DBGMGR	STC
(z/OS Explorer) JES Job Monitor	JMON	STC
(z/OS Explorer) RSE daemon	RSED	STC
(z/OS Explorer) RSE thread pool	RSEDx	OMVS
(ISPF) Interactive ISPF Gateway (TSO Commands serverice)	<userid>	JES
(ISPF) Legacy ISPF Gateway (TSO Commands service and SCLMDT)	<userid>x	OMVS
CARMA (batch)	CRA<port>	JES
CARMA (crastart)	<userid>x	OMVS
MVS build (batch job)	*	JES
z/OS UNIX build (shell commands)	<userid>x	OMVS
z/OS UNIX shell	<userid>	OMVS

Considerations for goal selection

The following general WLM considerations can help you to properly define the correct goal definitions for Developer for z Systems:

- You should base goals on what can actually be achieved, not what you want to happen. If you set goals higher than necessary, WLM moves resources from lower importance work to higher importance work which might not actually need the resources.
- Limit the amount of work assigned to the SYSTEM and SYSSTC service classes, because these classes have a higher dispatching priority than any WLM managed class. Use these classes for work that is of high importance but uses little CPU.

- Work that falls through the classification rules ends up in the SYSOTHER class, which has a discretionary goal. A discretionary goal tells WLM to just do the best it can when the system has spare resources.

When using response time goals:

- There must be a steady arrival rate of tasks (at least 10 tasks in 20 minutes) for WLM to properly manage a response time goal.
- Use average response time goals only for well controlled workloads, because a single long transaction has a big impact on the average response time and can make WLM overreact.

When using velocity goals:

- You usually cannot achieve a velocity goal greater than 90% for various reasons. For example, all the SYSTEM and SYSSTC address spaces have a higher dispatching priority than any velocity-type goal.
- WLM uses a minimum number of (using and delay) samples on which to base its velocity goal decisions. So the less work running in a service class, the longer it will take to collect the required number of samples and adjust the dispatching policy.
- Reevaluate velocity goals when you change your hardware. In particular, moving to fewer, faster processors requires changes to velocity goals.

OMVS

All workloads use the client user ID as base for the address space name. (z/OS UNIX will substitute "x" in the "Task Name" column by a random 1-digit number.)

The workloads will all end up in the same service class due to a common address space naming convention. You should specify a multi-period goal for this service class. The first periods should be high-performance, percentile response time goals, while the last period should have a moderate-performance velocity goal. Some workloads, such as the ISPF Client Gateway, will report individual transactions to WLM, while others do not.

<i>Table 5. WLM workloads - OMVS</i>		
Description	Task name	Workload
Legacy ISPF Gateway (TSO Commands service and SCLMDT)	<userid>x	OMVS
CARMA (crastart)	<userid>x	OMVS
CARMA (ISPF Client Gateway)	<userid> and <userid>x	OMVS
z/OS UNIX build (shell commands)	<userid>x	OMVS
z/OS UNIX shell	<userid>	OMVS

- Legacy ISPF Gateway

The Legacy ISPF Gateway is an ISPF service invoked by Developer for z Systems to execute non-interactive TSO and ISPF commands. This includes explicit commands issued by the client as well as implicit commands issued by the SCLMDT component of Developer for z Systems. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

- CARMA

CARMA is an optional Developer for z Systems server that is used to interact with host based Software Configuration Managers (SCMs), such as CA Endeavor® SCM. Developer for z Systems allows for different startup methods for a CARMA server, some of which become an OMVS workload. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

- z/OS UNIX build

When a client initiates a build for a z/OS UNIX project, z/OS UNIX REXEC (or SSH) will start a task that executes a number of z/OS UNIX shell commands to perform the build. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be moderate to substantial, depending on the size of the project.

- z/OS UNIX shell

This workload processes z/OS UNIX shell commands that are issued by the client. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

JES

JES-managed batch processes are used in various manners by Developer for z Systems. The most common usage is for MVS builds, where a job is submitted and monitored to determine when it ends. But Developer for z Systems could also start a CARMA server in batch, and communicate with it using TCP/IP.

<i>Table 6. WLM workloads - JES</i>		
Description	Task name	Workload
CARMA (batch)	CRA<port>	JES
MVS build (batch job)	*	JES

- CARMA

CARMA is a Developer for z Systems server that is used to interact with host based Software Configuration Managers (SCMs), such as CA Endevor[®] SCM. Developer for z Systems allows for different startup methods for a CARMA server, some of which become a JES workload. You should specify a high-performance, one-period velocity goal, because the task does not report individual transactions to WLM. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

- MVS build

When a client initiates a build for an MVS project, Developer for z Systems will start a batch job to perform the build. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be moderate to substantial, depending on the size of the project. Different moderate-performance goal strategies can be advisable, depending on your local circumstances.

- You could specify a multi-period goal with a percentile response time period and a trailing velocity period. In this case, your developers should be using mostly the same build procedure and similar sized input files to create jobs with uniform response times. There must also be a steady arrival rate of jobs (at least 10 jobs in 20 minutes) for WLM to properly manage a response time goal.
- A velocity goal is best suited for most batch-jobs, because this goal can handle highly variable execution times and arrival rates.

Chapter 5. Push-to-client considerations

Push-to-client, or host-based client control, supports central management of the following things:

- Client configuration files
- Client product version
- Project definitions

The following topics are covered in this chapter:

- [“Introduction” on page 23](#)
- [“Host-based projects” on page 23](#)

Developer for z Systems builds on top of IBM Explorer for z/OS. For z/OS Explorer related information, see "Push-to-client considerations" in the *IBM Explorer for z/OS Host Configuration Reference*.

Introduction

Developer for z Systems clients can pull client configuration files and product update information from the host when they connect, ensuring that all clients have common settings and that they are up-to-date.

The client administrator can create multiple client configuration sets and multiple client update scenarios to fit the needs of different developer groups. This allows users to receive a customized setup, based on criteria like membership of an LDAP group or permit to a security profile.

z/OS Projects can be defined individually through the z/OS Projects perspective on the client, or z/OS Projects can be defined centrally on the host and propagated to the client on an individual user basis. These "host-based projects" look and function exactly like projects defined on the client except that their structure, members, and properties cannot be modified by the client, and they are accessible only when connected to the host.

A development project manager defines a project and assigns individual developers to it.

See the [Developer for z Systems IBM Knowledge Center \(http://www.ibm.com/support/knowledgecenter/SSQ2R2/rdz_welcome.html\)](http://www.ibm.com/support/knowledgecenter/SSQ2R2/rdz_welcome.html) for details about how the development project manager can perform the tasks assigned to them.

When enabling configuration or version control support for multiple developer groups, one additional team will be involved in managing push-to-client. Which team this is depends on the option chosen to identify the groups a developer belongs to:

- An LDAP administrator maintains group definitions that place each developer in none, one, or more FEL . PTC . * LDAP groups.
- A security administrator maintains access lists to FEL . PTC . * security profiles. A developer can be authorized to none, one, or more profiles.

Host-based projects

z/OS Projects can be defined individually through the z/OS Projects perspective on the client, or z/OS Projects can be defined centrally on the host and propagated to the client on an individual user basis. These "host-based projects" look and function exactly like projects defined on the client except that their structure, members, and properties cannot be modified by the client, and they are only accessible when connected to the host.

The base directory for host-based projects is defined (by the client administrator) in `/var/zexpl/pushtoclient/keymapping.xml`, and is `/var/zexpl/pushtoclient/projects` by default.

To configure host-based projects, the project manager or lead developer needs to define the following types of configuration files. All files are UTF-8 encoded XML files.

- Project instance files are specific to a single user ID and point to reusable project definition files. Each user who works with host-based projects needs a subdirectory, `/var/zexpl/pushtoclient/projects/<userid>/`, containing one project instance file (`*.hbpin`) for each project to be downloaded.
- Project definition files define the structure and contents of the project and can be reused by multiple users. Project definition files (`*.hbppd`) list the subprojects contained by the project and are located in the root project definition directory or one of its subdirectories.
- Subproject definition files define the structure and contents of the subproject and can be reused by multiple users. Subproject definition files (`*.hbpsd`) define the set of resources required to build a single load module and are located in the root project definition directory or one of its subdirectories.
- Subproject properties files are properties files with variable substitution support and can be reused by multiple subprojects. Subproject property files (`*.hbppr`) support variable substitution to allow sharing of property files among multiple users and are located in the root project definition directory or one of its subdirectories.

Host-based projects are also eligible to participate in the multiple group setup. This eligibility means that host-based projects can be defined also in `/var/zexpl/pushtoclient/grouping/<devgroup>/projects/`.

When a workspace is bound to a specific group, and there are project definitions for a user in this group and in the default group, the user receives the project definitions from both the default and the specific group.

Chapter 6. CICSTS considerations

This chapter groups references to Developer for z Systems components that can work inside CICSTS regions.

xUnit support for CICS applications

For more information on zUnit CICS support, see the section "[xUnit support for CICS applications](#)" in chapter "Other customization tasks" of the *Host Configuration Guide*.

Bidirectional language support

For more information on Bidirectional language support, see the section "CICS bidirectional language support" in chapter "Other customization tasks" of the *Host Configuration Guide*.

Diagnostic IRZ messages for Enterprise Service Tools

For more information about diagnostic IRZ messages for Enterprise Service Tools, see the section "Diagnostic IRZ messages for Enterprise Service Tools" in chapter "Other customization tasks" of the *Host Configuration Guide*.

Chapter 7. SMF considerations

Shops can use Systems Management Facilities (SMF) records for purposes such as performance management, capacity planning, auditing, and accounting. Developer for z Systems provides SMF record type 122 subtype 1.

SMF type 122, subtype 1

SMF record type 122, subtype 1, always has a header section that is followed by 1 or more data sections:

- [“Header section” on page 27](#): Generic header for this SMF record
- [“Data section 1, Creator ID” on page 30](#): Identify which server created the SMF record
- [“Data section 2, Server initialization” on page 30](#): Holds server initialization information
- [“Data section 3, VU license handler status” on page 31](#): Holds information on activation code request processing
- [“Data section 4, Client UUID” on page 32](#): Holds unique client identifier
- [“Data section 5, Client labels” on page 32](#): Holds labels that describe additional client data
- [“Data section 6, Client data” on page 33](#): Holds additional data that describes the client.

Each data section can be present 0, 1, or more times. If there are multiple copies of the data section, then all sections are consecutive.

Client activation code request

Developer for z Systems clients require an activation code to enable all features. With VU-based licensing, the z/OS based server is capable of providing an activation code with a limited life span when a client requests an activation code, thus enabling the client upon connect. Developer for z Systems creates a SMF type 122, subtype 1 record each time a client requests an activation code.

Refer to section "Product enablement in IFAPRDxx" in the *Host Configuration Guide* to learn if your Developer for z Systems host is enabled to provide activation codes.

SMF record type 122, subtype 1, has a product header and up to 6 data sections when used for an activation code request:

- [“Header section” on page 27](#): Generic header for this SMF record
- [“Data section 1, Creator ID” on page 30](#): Identify which server created the SMF record
- [“Data section 2, Server initialization” on page 30](#): Holds server initialization information
- [“Data section 3, VU license handler status” on page 31](#): Holds information on activation code request processing
- [“Data section 4, Client UUID” on page 32](#): Holds unique client identifier
- [“Data section 5, Client labels” on page 32](#): Holds labels that describe additional client data
- [“Data section 6, Client data” on page 33](#): Holds additional data that describes the client.

Header section

The header section for SMF record type 122, subtype 1, holds a standard SMF header followed by fields that reference the various data sections. Character strings are in EBCDIC, left aligned and padded with blanks to fill the field.

Table 7. SMF record type 122 subtype 1, header section

Offsets	Type/Value	Length	Name	Description
0 (x0)	Structure	24	SMFHDR	** Standard SMF header **
0 (x0)	Structure	4	SMFRDW	** SMF Record descriptor word **
0 (x0)	Unsigned	2	SMFLEN	Record length. This field and the next field (total of 4 bytes) form the record descriptor word (RDW), which is removed when the record is dumped.
2 (x2)	Unsigned	2	SMFSEG	Spanned segment descriptor. This field and the previous field (total of 4 bytes) form the record descriptor word (RDW), which is removed when the record is dumped.
4 (x4)	Bitstring	1	SMFFLG	SMF header flag byte.
	.1..		SMSSTV	(x40) Subtypes are being used.
5 (x5)	Unsigned	1	SMFRTY	Record type: 122 (x7A)
6 (x6)	Unsigned	4	SMFTME	Time since midnight, in hundredths of a second, that has elapsed since the record was moved into the SMF buffer (format: HHMMSSth).
10 (xA)	Packed	4	SMFDAT	Date when the record was moved into the SMF buffer. In the form of 0cyydddF, where c is 0 for 19xx and 1 for 20xx, yy is the current year (0-99), ddd is the current day (1-366), and F is the sign.
14 (xE)	EBCDIC	4	SMFSID	System identification (from the SMF SID parameter).
18 (x12)	EBCDIC	4	SMFWID	Subsystem ID.
22 (x16)	Unsigned	2	SMFSTP	Record subtype: 1 (x1)
24 (x18)				
24 (x18)	Structure	40	SMF122t1h_Head	** SMF type 122 subtype 1 specific header **
24 (x18)		4		-- header description
24 (x18)	Unsigned	2	SMF122t1h_Len	SMF type 122 subtype 1 specific header length. It is set to the size of SMF122t1h_Head.
26 (x1A)	Unsigned	2	SMF122t1h_Cnt	Number of data section triplets. It is set to the number of data section identifier blocks that follow.
28 (x1C)	Structure	6		-- triplet for data section 1 (creator ID)
28 (x1C)	Unsigned	2	SMF122t1h_s1Len	Section length. It is set to the size of SMF122t1s1_Section.

Table 7. SMF record type 122 subtype 1, header section (continued)

Offsets	Type/Value	Length	Name	Description
30 (x1E)	Unsigned	2	SMF122t1h_S1Cnt	Number of sections. This field documents how many times this section is present in the record (can be 0). If the number is greater than 1, then all sections are consecutive.
32 (x20)	Unsigned	2	SMF122t1h_S1Offset	Section offset. The offset, from the start of the record, of this section block.
34 (x22)	Structure	6		-- triplet for data section 2 (server initialization)
34 (x22)	Unsigned	2	SMF122t1h_S2Len	Section length. It is set to the size of SMF122t1s2_Section.
36 (x24)	Unsigned	2	SMF122t1h_S2Cnt	Number of sections. This field documents how many times this section is present in the record (can be 0). If the number is greater than 1, then all sections are consecutive.
38 (x26)	Unsigned	2	SMF122t1h_S2Offset	Section offset. The offset, from the start of the record, of this section block.
40 (x28)	Structure	6		-- triplet for data section 3 (VU license handler)
40 (x28)	Unsigned	2	SMF122t1h_S3Len	Section length. It is set to the size of SMF122t1s3_Section.
42 (x2A)	Unsigned	2	SMF122t1h_S3Cnt	Number of sections. This field documents how many times this section is present in the record (can be 0). If the number is greater than 1, then all sections are consecutive.
44 (x2C)	Unsigned	2	SMF122t1h_S3Offset	Section offset. The offset, from the start of the record, of this section block.
46 (x2E)	Structure	6		-- triplet for data section 4 (client UUID)
46 (x2E)	Unsigned	2	SMF122t1h_S4Len	Section length. It is set to the size of SMF122t1s4_Section.
48 (x30)	Unsigned	2	SMF122t1h_S4Cnt	Number of sections. This field documents how many times this section is present in the record (can be 0). If the number is greater than 1, then all sections are consecutive.
50 (x32)	Unsigned	2	SMF122t1h_S4Offset	Section offset. The offset, from the start of the record, of this section block.
52 (x34)	Structure	6		-- triplet for data section 5 (client labels)

Offsets	Type/Value	Length	Name	Description
52 (x34)	Unsigned	2	SMF122t1h_S5Len	Section length. It is set to the size of SMF122t1s5_Section.
54 (x36)	Unsigned	2	SMF122t1h_S5Cnt	Number of sections. This field documents how many times this section is present in the record (can be 0). If the number is greater than 1, then all sections are consecutive.
56 (x38)	Unsigned	2	SMF122t1h_S5Offset	Section offset. The offset, from the start of the record, of this section block.
58 (x3A)	Structure	6		-- triplet for data section 6 (client data)
58 (x3A)	Unsigned	2	SMF122t1h_S6Len	Section length. It is set to the size of SMF122t1s6_Section.
60 (x3C)	Unsigned	2	SMF122t1h_S6Cnt	Number of sections. This field documents how many times this section is present in the record (can be 0). If the number is greater than 1, then all sections are consecutive.
62 (x3E)	Unsigned	2	SMF122t1h_S6Offset	Section offset. The offset, from the start of the record, of this section block.

Data section 1, Creator ID

Data section 1 for SMF record type 122, subtype 1, identifies the server and the function that created this SMF record. Character strings are in EBCDIC, left aligned and padded with blanks to fill the field.

Offsets	Type/Value	Length	Name	Description
0 (x0)	Structure	20	SMF122t1s1_Section	** section 1 (creator ID) **
0 (x0)	EBCDIC	8	SMF122t1s1_Plex	Sysplex name
8 (x8)	EBCDIC	8	SMF122t1s1_Sys	System name
16 (x10)	Unsigned	2	SMF122t1s1_Server	Server ID. This is set to the RSED port number.
18 (x12)	Bitstring	2	SMF122t1s1_Flags	Flags
1		SMF122t1s1F_Lic	(x1) VU license handler created this record.

Data section 2, Server initialization

Data section 2 for SMF record type 122, subtype 1, holds system and server initialization information. Character strings are in EBCDIC, left aligned and padded with blanks to fill the field. It is created in these situations:

- The first time a client requests an activation code since server startup.

- The first time a client requests an activation code since dynamic changes to SYS1.PARMLIB(IFAPRDxx) that impact Developer for z Systems.

Table 9. SMF record type 122 subtype 1, data section 2

Offsets	Type/Value	Length	Name	Description
0 (x0)	Structure	22	SMF122t1s2_Section	** section 2 (server initialization) **
0 (x0)	Unsigned	4	SMF122t1s2_iTime	IPL time stamp. Time since midnight, in hundredths of a second, that has elapsed when the system was IPL'd (format: HHMMSSth).
4 (x4)	Packed	4	SMF122t1s2_iDate	IPL date stamp. Date when the system was IPL'd. In the form of 0cyydddF, where c is 0 for 19xx and 1 for 20xx, yy is the current year (0-99), ddd is the current day (1-366), and F is the sign.
8 (x8)	EBCDIC	8	SMF122t1s2_ProdID	Registered Product ID (PID). This is set to the PID used by the server to (re-)register itself, following the rules defined in SYS1.PARMLIB(IFAPRDxx).
16 (x10)	Unsigned	1	SMF122t1s2_Ver	Server version. This is set to the first nibble of environment variable \$IDZ_VERSION.
17 (x11)	Unsigned	1	SMF122t1s2_Rel	Server release. This is set to the second nibble of environment variable \$IDZ_VERSION.
18 (x12)	Unsigned	1	SMF122t1s2_Mod	Server modification. This is set to the third nibble of environment variable \$IDZ_VERSION.
19 (x13)	Unsigned	1	SMF122t1s2_Lvl	Server level. This is set to the fourth nibble of environment variable \$IDZ_VERSION.
20 (x14)	Bitstring	2	SMF122t1s2_Flags	Flags
			(none)

Data section 3, VU license handler status

Data section 3 for SMF record type 122, subtype 1, holds information pertinent to how the VU license handler processed the activation code request by a client. Character strings are in EBCDIC, left aligned and padded with blanks to fill the field.

Table 10. SMF record type 122 subtype 1, data section 3

Offsets	Type/Value	Length	Name	Description
0 (x0)	Structure	4	SMF122t1s3_Section	** section 3 (VU license handler) **
0 (x0)	Bitstring	2	SMF122t1s3_Flags	Flags
1		SMF122t1s3F_VUon	(x0001) Client activation code provided.

Offsets	Type/Value	Length	Name	Description
2 (x2)	Unsigned	2	SMF122t1s3_Track	Request tracker. This field increments by one each time a request for an activation code is processed. The counter wraps back to 0 when the maximum value is reached.

Data section 4, Client UUID

Data section 4 for SMF record type 122, subtype 1, holds the Universally Unique Identifier (UUID) provided by the client during the request for an activation code. The length of this UUID can vary, and is documented in field SMF122t1h_S4Len of the header section. Character strings are in UTF8.

Offsets	Type/Value	Length	Name	Description
0 (x0)	Structure	?	SMF122t1s4_Section	** section 4 (client UUID) **
0 (x0)	UTF8	?	SMF122t1s4_UUID	Unique client ID

Data section 5, Client labels

Data section 5 for SMF record type 122, subtype 1, holds a list of comma (,) separated labels that describe the data elements in data section 6, client data. The client provides most of this data during the request for an activation code. The length of this string can vary, and is documented in field SMF122t1h_S5Len of the header section. Character strings are in UTF8.

- productName: Name of the Developer for z Systems client
- productVersion: Version of the Developer for z Systems client
- productRelease: Release of the Developer for z Systems client
- productModification: Modification of the Developer for z Systems client
- zexplIPAddress: IP address of the Developer for z Systems client
- zexplHostName: DNS name of the Developer for z Systems client
- zexplUserId: User ID of the person that is using the Developer for z Systems client
- productAPIVersionClient: Version of the activation code API used by the Developer for z Systems client
- zexplAPIVersionClient: Version of the activation code API used by the z/OS Explorer client
- zexplAPIVersionHost: Version of the activation code API used by the z/OS Explorer server
- productAPIVersionHost: Version of the activation code API used by the Developer for z Systems server
- returnCode: Result of z/OS Explorer validating correctness of the data it transferred on behalf of Developer for z Systems

Offsets	Type/Value	Length	Name	Description
0 (x0)	Structure	?	SMF122t1s5_Section	** section 5 (client labels) **
0 (x0)	UTF8	?	SMF122t1s5_Labels	Client data labels in comma separated list

Data section 6, Client data

Data section 6 for SMF record type 122, subtype 1, holds a list of comma (,) separated data elements, which are described in data section 5, client labels. The client provides most of this data during the request for an activation code. The length of this string can vary, and is documented in field SMF122t1h_S6Len of the header section. Character strings are in UTF8. If the data itself holds a comma (,) it is replaced by its Unicode representation, `,`. Other characters that are replaced by their Unicode representation are the semicolon (;), `;`, and the tilde (~), `~`.

Offsets	Type/Value	Length	Name	Description
0 (x0)	Structure	?	SMF122t1s6_Section	** section 6 (client data) **
0 (x0)	UTF8	?	SMF122t1s6_Client	Client data in comma separated list

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide 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:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

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 document 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.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains 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.

Programming interface information

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.



SC27-8578-03

