IBM Rational Developer for System z
Version 8.0.3

Host Configuration Reference Guide

SC14-7290-01

IBM Rational Developer for System z
Version 8.0.3

Host Configuration Reference Guide

SC14-7290-01

Note

Before using this document, read the general information under "Documentation notices for IBM Rational Developer for System z" on page 185.

Second edition (October, 2011)

This edition applies to IBM Rational Developer for System z Version 8.0.3 (program number 5724-T07) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications by phone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. eastern standard time (EST). The phone number is (800) 879-2755. The fax number is (800) 445-9269. Faxes should be sent Attn: Publications, 3rd floor.

You can also order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

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.

Contents

Figures

Tables

About this document

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

From here on, the following names are used in this manual:

- *IBM Rational Developer for System z* is called *Developer for System z*.
- *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*.

For earlier releases, including IBM WebSphere Developer for System z, IBM WebSphere Developer for zSeries and IBM WebSphere Studio Enterprise Developer, use the configuration information found in the Host Configuration Guide and Program Directories for those releases.

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

- *Rational Developer for System z Host Configuration Guide* (SC23-7658) describes in detail all planning tasks, configuration tasks and options (including optional ones) and provides alternative scenarios.
- *Rational Developer for System z Host Configuration Reference* (SC14-7290) describes Developer for System z design and gives background information for various configuration tasks of Developer for System z, z/OS components, and other products (such as WLM and CICS) related to Developer for System z.
- *Rational Developer for System z Host Configuration Quick Start Guide* (GI11-9201) describes a minimal setup of Developer for System z.
- *Rational Developer for System z Host Configuration Utility* (SC14-7282) describes the Host Configuration Utility, an ISPF panel application that guides you through basic and common optional customization steps for Developer for System z.

The information in this document applies to all Rational Developer for System z Version 8.0.3 packages including IBM Rational Developer for zEnterprise™.

Who should use this document

This document is intended for system programmers configuring and tuning IBM Rational Developer for System z Version 8.0.3.

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

Summary of changes

This section summarizes the changes for *IBM Rational Developer for System z Version 8.0.3 Host Configuration Reference*, SC14-7290-01 (updated October 2011).

Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This document contains information previously presented in *IBM Rational Developer for System z Version 8.0.1 Host Configuration Reference*, SC14-7290-00.

New information:

- Updated z/OS UNIX directory structure. See "z/OS UNIX directory structure" on page 11.
- Added information on host-based client control. See Chapter 7, "Push-to-client considerations," on page 99.
- Added security-related push-to-client information. See "Push-to-client developer groups" on page 28.
- Document usage of Managed ACEEs. See "Managed ACEE" on page 28.
- Added information on automated audit log processing. See "Audit processing" on page 19.
- Updated information on security and audit related directives in configuration files. See "Developer for System z configuration files" on page 30.
- Added additional TCP/IP information. See Chapter 3, "TCP/IP considerations," on page 47.
- Updated Certificate Authority information for SSL communication. See Appendix A, "Setting up SSL and X.509 authentication," on page 155.
- Updated resource usage. See "Resource usage" on page 63.

This document contains information previously presented in *IBM Rational Developer for System z Version* 7.6.1 Host Configuration Guide, SC23-7658-04.

New information:

- CARMA section in Understanding Developer for System z. See "CARMA" on page 7.
- General TCP/IP related information. See Chapter 3, "TCP/IP considerations," on page 47.
- B37 space abend resolution. See "Error feedback B37 space abend" on page 153.

Removed information:

- The information previously presented in *IBM Rational Developer for System z Version 7.6.1 Host Configuration Guide* (SC23-7658-04) is now split into two documents: *IBM Rational Developer for System z Host Configuration Guide* (SC23-7658) and *IBM Rational Developer for System z Host Configuration Reference* (SC14-7290).
- Information regarding APPC setup has moved to white paper *Using APPC to provide TSO command services* (SC14-7291).
- Setting up INETD

Description of document content

This section summarizes the information presented in this document.

Understanding Developer for System z

The Developer for System z 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.

Security considerations

Developer for System z provides mainframe access to users on a non-mainframe workstation. Validating connection requests, providing secure communication between the host and the workstation, and authorizing and auditing activity are therefore important aspects of the product configuration.

TCP/IP considerations

Developer for System z 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.

WLM considerations

Unlike traditional z/OS applications, Developer for System z is not a monolithic application that can be identified easily to Workload Manager (WLM). Developer for System z consists of several components that interact to give the client access to the host services and data. Some of these services are active in different address spaces, resulting in different WLM classifications.

Tuning considerations

RSE (Remote Systems Explorer) is the core of Developer for System z. To manage the connections and workloads from the clients, RSE is composed of a daemon address space, which controls thread pooling address spaces. The daemon acts as a focal point for connection and management purposes, while the thread pools process the client workloads.

This makes RSE a prime target for tuning the Developer for System z setup. However, maintaining hundreds of users, each using 16 or more threads, a certain amount of storage, and possibly one or more address spaces requires proper configuration of both Developer for System z and z/OS.

Performance considerations

z/OS is a highly customizable operating system, and (sometimes small) system changes can have a huge impact on the overall performance. This chapter highlights some of the changes that can be made to improve the performance of Developer for System z.

Push-to-client considerations

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

- Client configuration files
- Client product version
- Project definitions

CICSTS considerations

This chapter contains information useful for a CICS Transaction Server administrator.

Customizing the TSO environment

This chapter assists you with mimicking a TSO logon procedure by adding DD statements and data sets to the TSO environment in Developer for System z.

Running multiple instances

There are times that you want multiple instances of Developer for System z active on the same system, for example, when testing an upgrade. However, some resources such as TCP/IP ports cannot be shared, so the defaults are not always applicable. Use the information in this chapter to plan the coexistence of the different instances of Developer for System z, after which you can use this configuration guide to customize them.

Troubleshooting configuration problems

This chapter is provided to assist you with some common problems that you may encounter during your configuration of Developer for System z, and has the following sections:

- Log and setup analysis using FEKLOGS
- Log files
- Dump files
- Tracing
- z/OS UNIX permission bits
- Reserved TCP/IP ports
- Address Space size
- APPC transaction and TSO Commands service
- Miscellaneous information

Setting up SSL and X.509 authentication

This appendix is provided to assist you with some common problems that you may encounter when setting up Secure Socket Layer (SSL), or during checking or modifying an existing setup. This appendix also provides a sample setup to support users authenticating themselves with an X.509 certificate.

Setting up TCP/IP

This appendix is provided to assist you with some common problems that you may encounter when setting up TCP/IP, or during checking or modifying an existing setup.

Chapter 1. Understanding Developer for System z

The Developer for System z 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

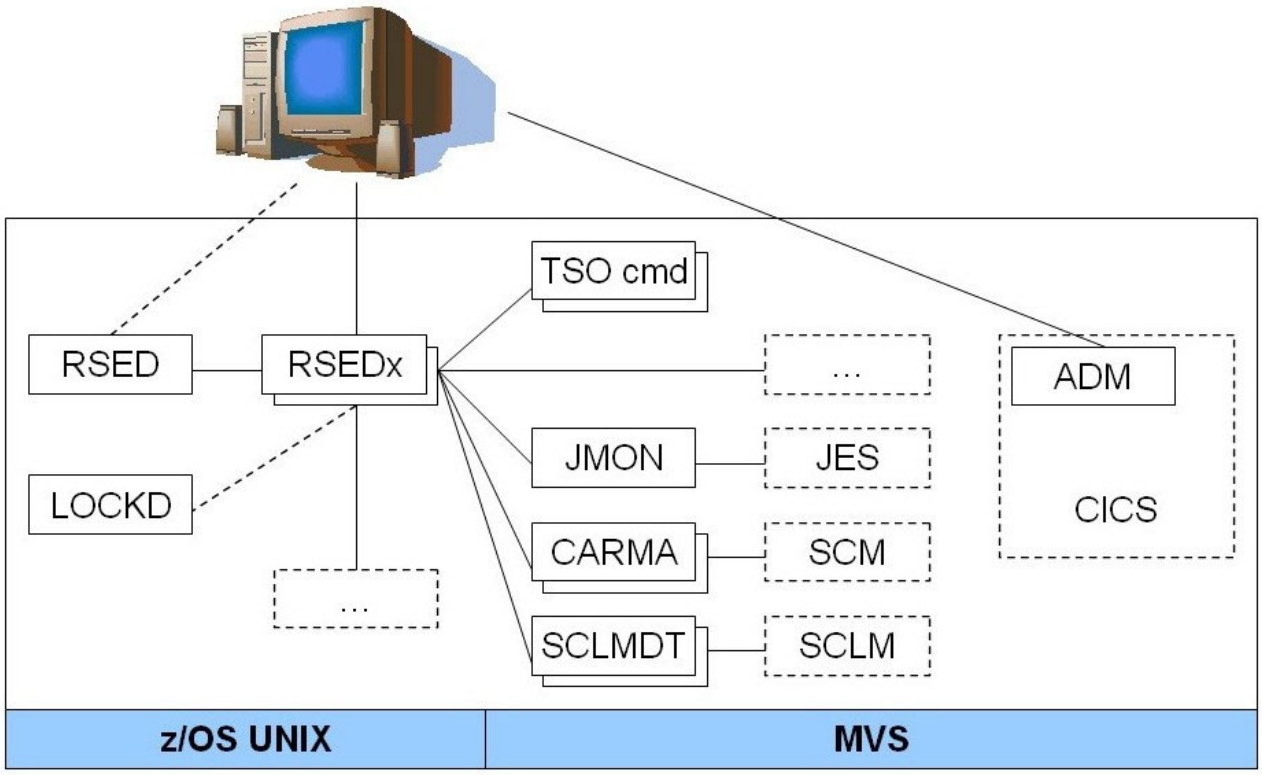


Figure 1. Component overview

Figure 1 shows a generalized overview of the Developer for System z layout on your host system.

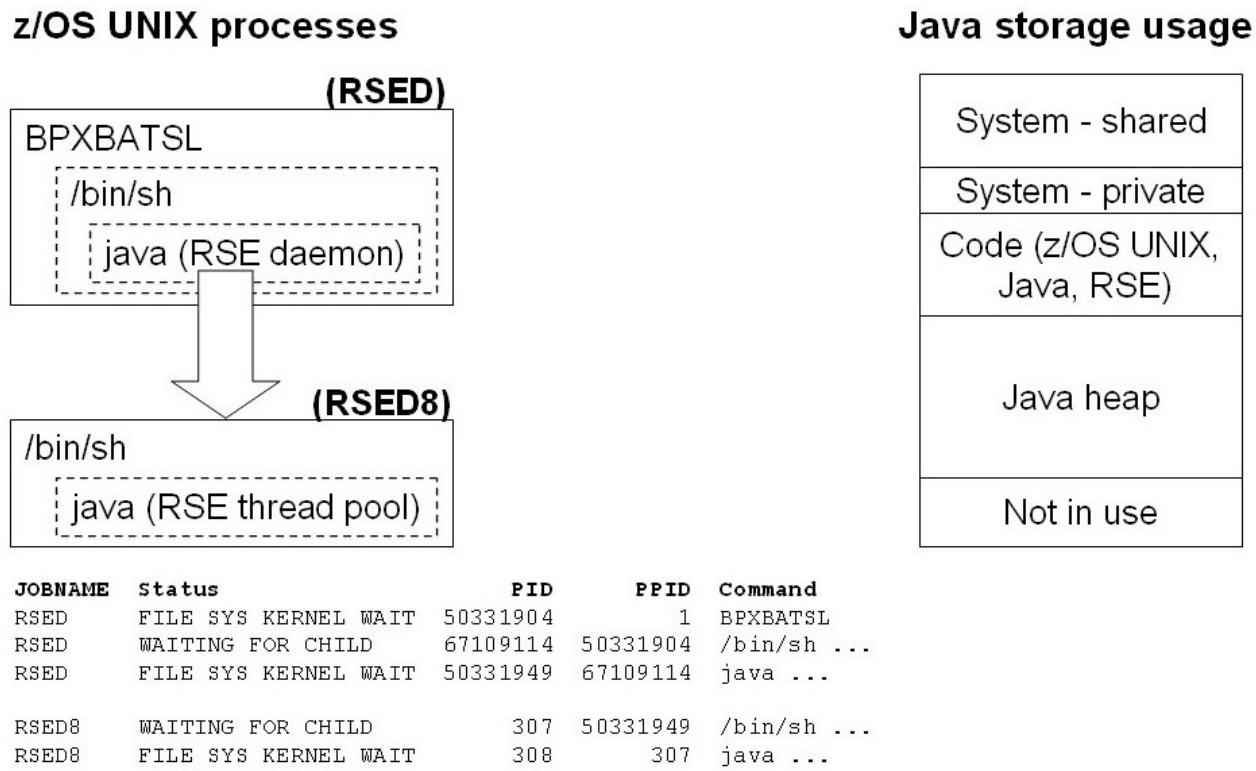
- Remote Systems Explorer (RSE) provides core services, such as connecting the client to the host and starting other servers for specific services. RSE consists of two logical entities:

  - 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.
- Lock Daemon (LOCKD) provides tracking services for data set locks.
- TSO Commands Service (TSO cmd) provides a batch-like interface for TSO and ISPF commands.
- 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 Endevor.
- SCLM Developer Toolkit (SCLMDT) provides an interface to enhance and interact with SCLM.
- Application Deployment Manager (ADM) provides various CICS related services.
- More services are available, which can be provided by Developer for System z 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 pooling 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 `rsed.envvars` configuration file, and the amount of actual client connections, multiple thread pool address spaces can be started by the daemon.

RSE as a Java application




```
JOBNAME  Status                    PID       PPID  Command
RSED     FILE SYS KERNEL WAIT  50331904          1  BPXBATSL
RSED     WAITING FOR CHILD     67109114   50331904  /bin/sh ...
RSED     FILE SYS KERNEL WAIT  50331949   67109114  java ...

RSED8    WAITING FOR CHILD           307   50331949  /bin/sh ...
RSED8    FILE SYS KERNEL WAIT        308        307  java ...
```

Figure 2. RSE as a Java application

Figure 2 shows a basic view of resource usage (processes and storage) by RSE.

RSE is a Java application, which means that it is active in the z/OS UNIX environment. This allows for easy porting to different host platforms and straightforward communication with the Developer for System z client, which is also a Java application (based on the Eclipse framework). Therefore, basic knowledge of how z/OS UNIX and Java work is very helpful when you try to understand Developer for System z.

In z/OS UNIX, a program runs in a process, which is identified by a PID (Process ID). Each program is active in its own process, so invoking another program creates a new process. The process that started a process is referenced with a PPID (Parent PID), the new process is called a child process. The child process can run in the same address space or it can be spawned (created) in a new address space. A new process that runs in the same address space can be compared to executing a command in TSO, while the spawning one in a new address space is similar to submitting a batch job.

Note that a process can be single- or multi-threaded. In a multi-threaded application (such as RSE), the different threads compete for system resources as if they were separate address spaces (with less overhead).

Mapping this process information to the RSE sample in Figure 2, we get the following flow:

1. When the RSED task is started, it executes BPXBATSL, which invokes z/OS UNIX and creates a shell environment – PID 50331904.
2. In this process, the `rsed.sh` shell script is executed, which runs in a separate process (/bin/sh) – PID 67109114.
3. The shell script sets the environment variables defined in `rsed.envvars` and executes Java with the required parameters to start the RSE daemon – PID 50331949.
4. RSE daemon will spawn off a new shell in a child process (RSED8) – PID 307.
5. In this shell, the environment variables defined in `rsed.envvars` are set and Java is executed with the required parameters to start the RSE thread pool – PID 308.

Java applications, such as RSE, do not allocate storage directly, but use Java memory management services. These services, like allocating storage, freeing storage, and garbage collection, work within the limits of the Java heap. The minimum and maximum size of the heap is defined (implicitly or explicitly) during Java startup.

This implies that getting the most out of the available address space size is a balancing act of defining a large heap size while leaving enough room for z/OS to store a variable amount of system control blocks (dependent on the number of active threads).

Task owners

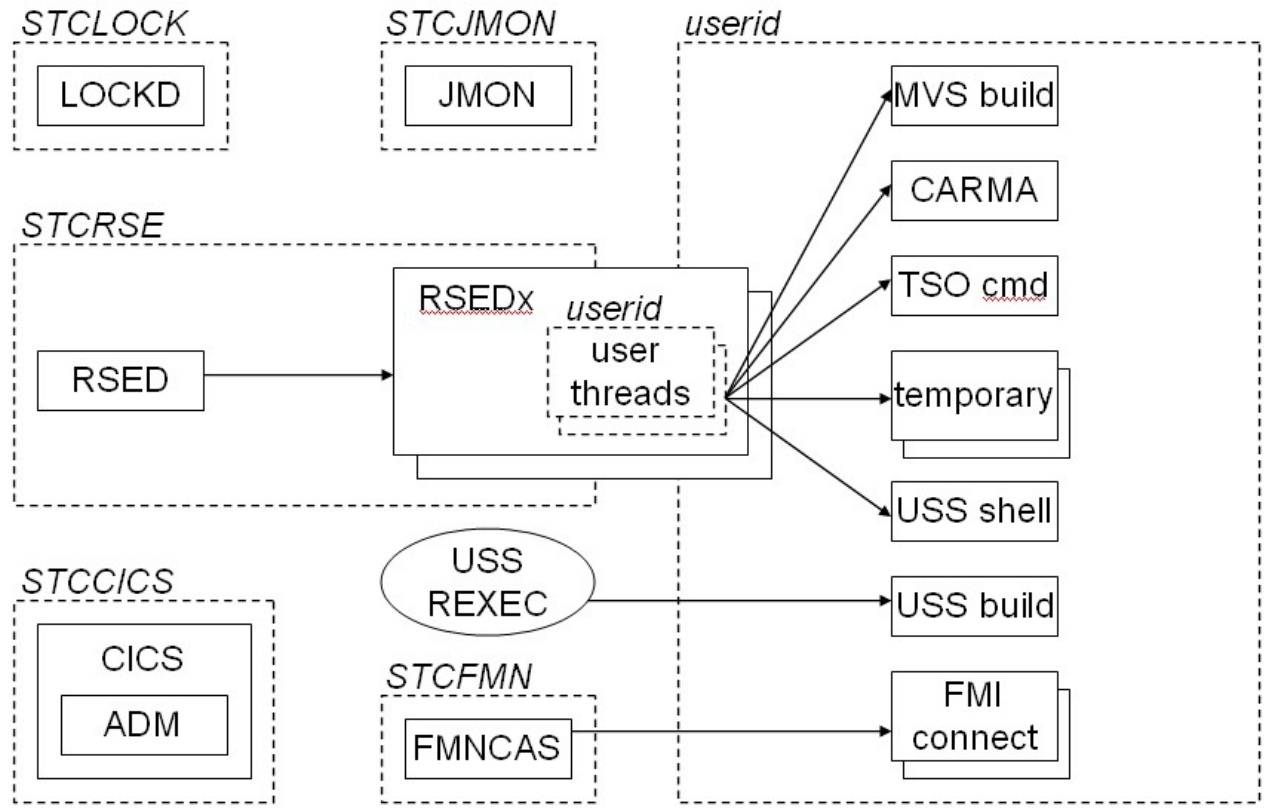


Figure 3. Task owners

Figure 3 on page 4 shows a basic overview of the owner of the security credentials used for various Developer for System z tasks.

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 3 on page 4 shows the Developer for system z started tasks (LOCKD, JMON and RSED), and sample started tasks and system services that Developer for System z communicates with. Application Deployment Manager (ADM) is active inside a CICS region. FMNCAS is the File Manager started task. 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, an APPC transaction, 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, File Manager will start a new address space for each data set (or member) it has to process on behalf of Developer for System z. 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 3 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.

Connection flow

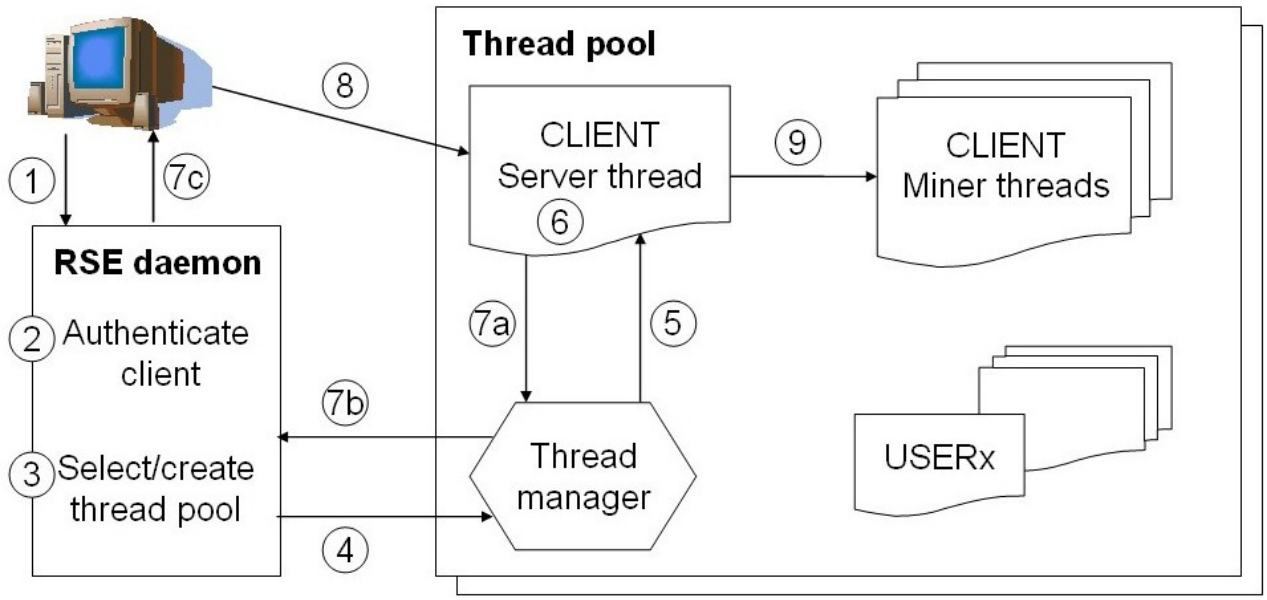


Figure 4. Connection flow

Figure 4 shows a schematic overview of how a client connects to the host using Developer for System z. It also briefly explains how PassTickets are used.

1. The client logs on to the daemon (port 4035).
2. RSE daemon authenticates the client, using the credentials presented by the client.
3. RSE daemon selects an existing thread pool or starts a new one if all are full.
4. RSE daemon passes the client user ID on to the thread pool.
5. The thread pool creates a client specific RSE server thread, using the client user ID and a PassTicket for authentication.
6. The client server thread binds to a port for future client communication.
7. The client server thread returns the port number for the client to connect to.
8. The client disconnects from RSE daemon and connects to the provided port number.
9. The client server thread starts other user specific threads (miners), always using the client user ID and a PassTicket for authentication. These threads provide the user-specific services requested by the client.

The previous description shows the thread-oriented design of RSE. Instead of starting an address space per user, multiple users are serviced by a single thread pool address space. Within the thread pool, each miner (a user specific service) is active in its own thread with the user's security context assigned to it, ensuring a secure setup. This design accommodates large number of users with limited resource usage, but does imply that each client will use multiple threads (16 or more, depending on the performed tasks).

From a network point of view, Developer for system z acts similar to FTP in passive mode. The client connects to a focal point (RSE daemon) and then drops

the connection and reconnects to a port number provided by the focal point. The following logic controls the selection of the port that is used for the second connection:

1. If the client specified a non-zero port number in the subsystem properties tab, then RSE server will use that port for the bind. If this port is not available, the connection fails.
2. If `_RSE_PORTRANGE` is specified in `rsed.envvars`, then RSE server will bind to a port from this range. If no port is available, the connection fails. Note that RSE server does not need the port exclusively for the duration of the client connection. It is only in the time span between the (server) bind and the (client) connect that no other RSE server can bind to the port. This means that most connections will be using the first port in the range, with the rest of the range being a buffer in case of multiple simultaneous logons.
3. If no limitations are set, RSE server will bind to port 0. The result is that TCP/IP chooses the port number.

The usage of PassTickets for all z/OS services that require authentication allows Developer for System z to invoke these services at will without storing the password or constantly prompting the user for it. Use of PassTickets for all z/OS services also allows for alternative authentication methods during logon, such as one-time passwords and X.509 certificates.

CARMA

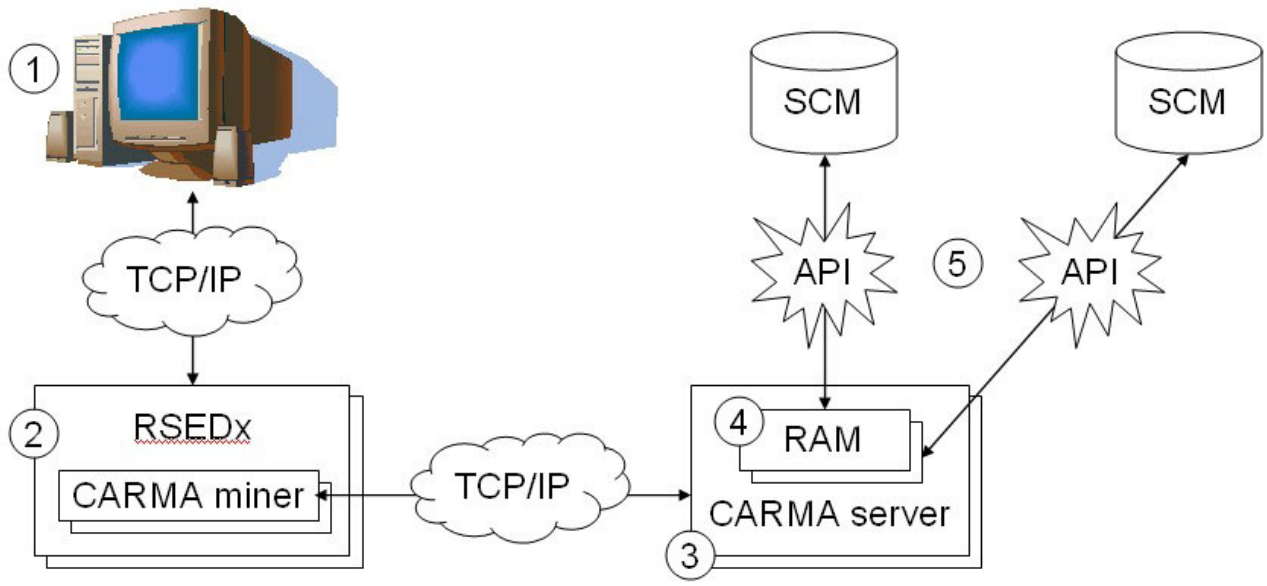


Figure 5. CARMA flow

CARMA (Common Access Repository Manager) is used to access a host based Software Configuration Manager (SCM), for example CA Endevor® SCM. Figure 5 shows a schematic overview of how a Developer for System z 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.
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 System z supports multiple methods to start a CARMA server. Each method has benefits and drawbacks. Developer for System z 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 System z uses this environment to run the CARMA server, `CRASERV`. Developer for System z 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 System z uses this environment to run the CARMA server, CRASERV. Developer for System z provides multiple CRASUB* members, each preconfigured for a specific RAM.

Lock daemon

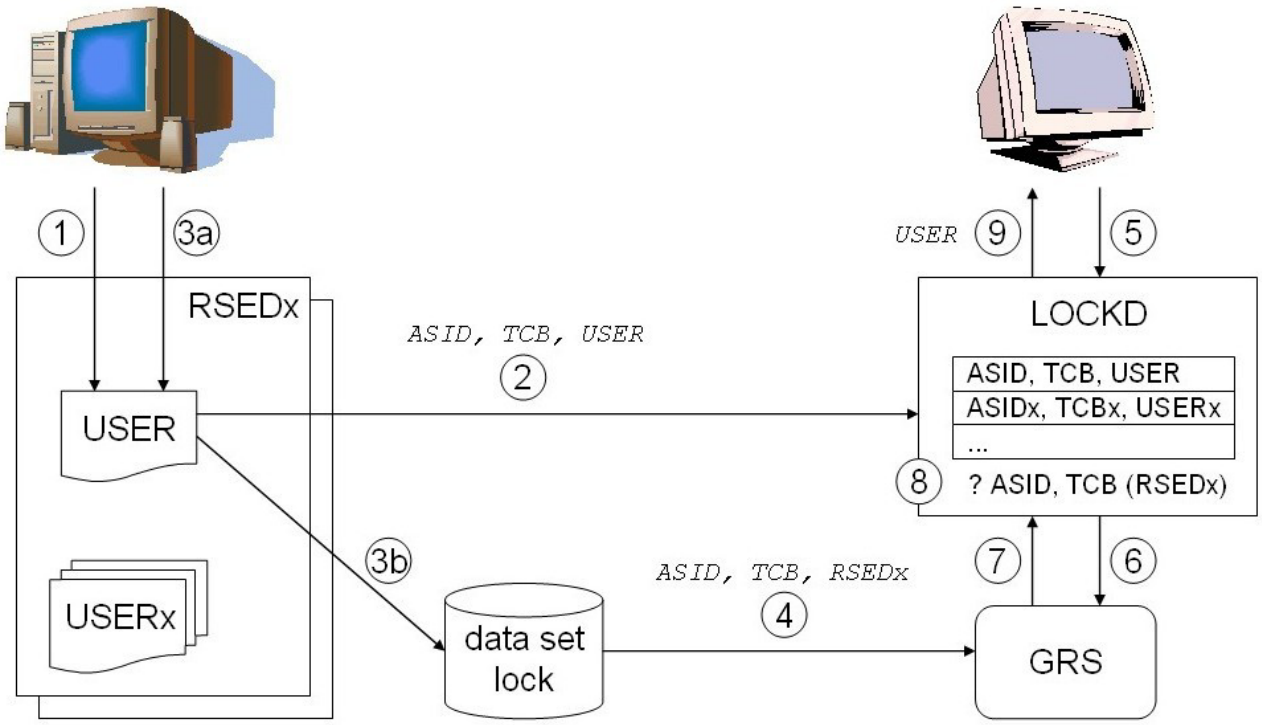


Figure 6. Lock daemon flow

Figure 6 shows a schematic overview of how the lock daemon determines which Developer for System z client owns a data set lock.

1. The client logs on, which creates a user-specific RSE server thread (USER) inside a thread pool (RSEDx).
2. RSE server registers a newly-connected user with the lock daemon. The registration information contains the Address Space Identifier (which is the ASID of the thread pool), the Task Control Block (TCB) identifier (user-specific), and the user ID.
3. The client opens a data set in edit, which instructs RSE server to get an exclusive lock on the data set.
4. The system registers the ASID, TCB and task name (RSEDx) of the requestor as part of lock process. This information is stored in the Global Resource Serialization (GRS) queues.
5. An operator (or RSE server on behalf of a client) queries the lock daemon for the lock status of the data set.
6. The lock daemon scans the GRS queues to learn if the data set is locked.
7. The daemon retrieves the ASID, TCB and task name of the lock owner.
8. The retrieved ASID and TCB are compared against the ASID and TCB combos of registered clients.
9. The related client user ID is returned to the requestor when a match is found. Otherwise, the task name retrieved from GRS is returned.

With the single-server setup of Developer for System z, where multiple users are assigned to a single thread pool address space, z/OS lost the ability to track who owns a lock on a data set or member. System commands stop at address space level, which is the thread pool.

To address this problem, Developer for System z provides the lock daemon. The lock daemon can track all dataset/member locks done by RSE users, as well as locks done by other products, such as ISPF.

RSE server registers a newly-connected user with the lock daemon. The registration information contains the Address Space Identifier (which is the ASID of the thread pool), the Task Control Block (TCB) ID (user-specific), and the user ID.

Note that registration is done at connect time only, so all RSE users active before the lock daemon was started (or restarted) will not be registered.

When the lock daemon receives a dataset query (by means of a modify query operator command or from the client by way of RSE server), the daemon scans the system's Global Resource Serialization (GRS) queues. If the ASID and TCB match that of a registered user, the user ID is returned as lock owner. Otherwise the jobname/user ID related to the ASID is returned as lock owner.

A console message (FEK513W) with the registration information is displayed if the registration fails. This allows an operator to match the values against the output of a **DISPLAY GRS,RES=(*,dataset*)** operator command in order to find the lock owner.

Note: Successful registrations are also listed in DD STDOUT of the server if `log_level` is set to `2`. This is useful to do the manual mapping for successful registrations that were removed after a restart of the lock daemon.

Freeing a lock

Under normal circumstances, a data set or member is locked when the client opens it in edit mode, and freed when the client closes the edit session.

Certain error conditions can prevent this mechanism from working as designed. In this case, the user holding the lock can be canceled using RSE's **modify cancel** operator command, as described in "Operator commands" in the *Host Configuration Guide* (SC23-7658). Active data set locks belonging to this user are freed during the process.

z/OS UNIX directory structure

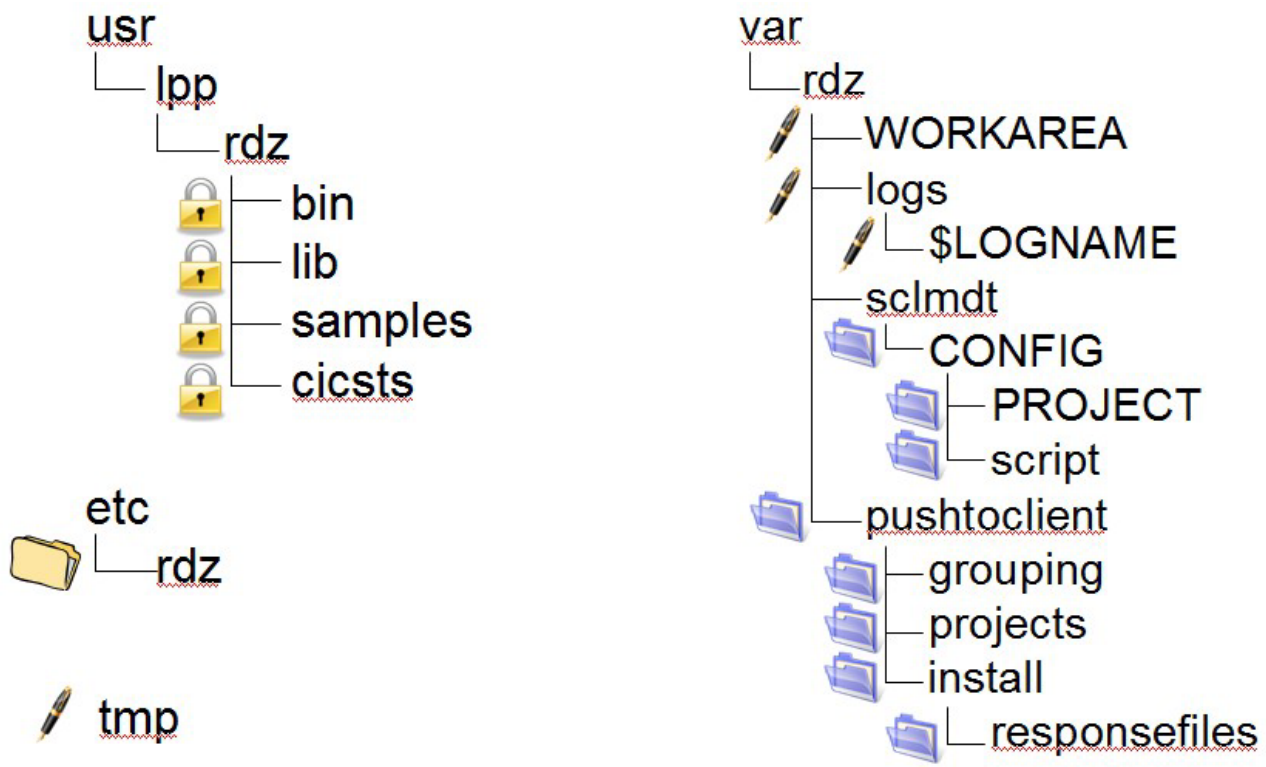


Figure 7. z/OS UNIX directory structure

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

- `/usr/lpp/rdz/` is the root path for the Developer for System z product code. The actual location is specified in the `RSED` and `LOCKD` started tasks (variable `HOME`). The files within are maintained by SMP/E.
- `/etc/rdz/` holds the RSE and miner-related configuration files. The actual location is specified in the `RSED` and `LOCKD` started tasks (variable `CNFG`). The files within are maintained by the system programmer.
- `/tmp/` is used by ISPF's TSO/ISPF Client Gateway and various miners to store temporary data. Some IVPs store their output here. The files within are maintained by ISPF, the miners, and the IVPs. The location can be customized with the `TMPDIR` variable in `rsed.envvars`. It is also the default location for Java dump files, which can be customized with the `_CEE_DUMPTARG` variable in `rsed.envvars`.

  Note: `/tmp/` requires permission bit mask `777` to allow each client to create temporary files.
- `/var/rdz/WORKAREA/` is used by ISPF's TSO/ISPF Client Gateway and SCLMDT to transfer data between z/OS UNIX and MVS based address spaces. The actual location is specified in `rsed.envvars` (variable `_CMDSERV_WORK_HOME`). The files within are maintained by ISPF and SCLMDT.

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

- /var/rdz/logs/ holds the logs of RSE daemon and RSE thread pool servers. The actual location is specified in rsed.envvars (variable daemon.log). The files within are maintained by RSE.
- /var/rdz/logs/$LOGNAME/ holds the user-specific logs of the RSE server and miners. The actual location is specified in rsed.envvars (variable user.log and DSTORE_LOG_DIRECTORY). The files within are maintained by RSE and the miners.

  Note: /var/rdz/logs/ requires permission bit mask 777 to allow each client to create his $LOGNAME directory and store the user-specific log files.
- /var/rdz/sclmdt/CONFIG/ holds general SCLMDT configuration files. The actual location is specified in rsed.envvars (variable SCLMDT_CONF_HOME). The files within are maintained by the SCLM administrator.
- /var/rdz/sclmdt/CONFIG/PROJECT/ holds SCLMDT project configuration files. The actual location is specified in rsed.envvars (variable SCLMDT_CONF_HOME). The files within are maintained by the SCLM administrator.
- /var/rdz/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/rdz/pushtoclient/ holds client configuration files, client product update information, and host-based project 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 System z client administrator.
- /var/rdz/pushtoclient/grouping/ holds group-specific client configuration files, client product update information, and host-based project information that is pushed to the client upon connection to the host. The actual location is specified in pushtoclient.properties (variable pushtoclient.folder plus suffix /grouping). The files within are maintained by a Developer for System z client administrator.
- /var/rdz/pushtoclient/projects/ holds the host-based project definition files. The actual location is specified in /var/rdz/pushtoclient/keymapping.xml, which is created and maintained by a Developer for System z client administrator. The files within are maintained by a project manager or lead developer.
- /var/rdz/pushtoclient/install/ holds configuration files used to update the client product version upon connection to the host. The actual location is specified in /var/rdz/pushtoclient/keymapping.xml, which is created and maintained by a Developer for System z client administrator. The files within are maintained by a client administrator.
- /var/rdz/pushtoclient/install/responsefiles/ holds configuration files used to update the client product version upon connection to the host. The actual location is specified in /var/rdz/pushtoclient/keymapping.xml, which is created and maintained by a Developer for System z client administrator. The files within are maintained by a client administrator.

Update privileges for non-system administrators

The data in /var/rdz/pushtoclient/ is maintained by non-system administrators, such as project managers, who might not have many update privileges in z/OS UNIX. Therefore, it is important to understand how z/OS UNIX sets access permissions during file creation to ensure you have workable but secure setup.

UNIX standards dictate that permissions can be set for three types of users: owner, group, and other. Read, write, and execute permissions can be set for each type individually.

z/OS UNIX sets the UID (user ID) and GID (group ID) to the following values when a file is created:

- The UID is set to the effective UID of the creating thread.
- The GID is set to the GID of the owning directory. If security profile `FILE.GROUPOWNER.SETGID` is defined in the `UNIXPRIV` class, then the effective GID of the creating thread is used by default instead. See *UNIX System Services Planning* (GA22-7800) for more details.

Each site can set their own default access permission mask, but a common mask allows read and write permission to the owner, and read permission to group and other.

Data in `/var/rdz/pushtoclient/` is created using the access permission mask defined in the `file.permission` directive of `pushtoclient.properties`. The default value allows read and write permission for owner and group, and read permission for other. All have execute permission. The final access permissions should allow read and execute for all, and write for the Developer for System z client administrators that maintain the data.

Data in `/var/rdz/pushtoclient/projects/` is created using no specific access permission mask. The final access permissions should allow read permission for all, and write permission for the project managers that maintain the data.

Useful security commands

To ensure that a group of project managers or Developer for System z client administrators can manage the data in these directories, your security administrator might have to create a group with a valid OMVS segment for them. This group is preferably the default group for the involved user IDs. Refer to *Security Server RACF® Command Language Reference* (SA22-7687) for more information on the following sample RACF commands:

```
ADDGROUP RDZPROJ OMVS(GID(1200))
CONNECT IBMUSER GROUP(RDZPROJ)
ALTUSER IBMUSER DFLTGRP(RDZPROJ)
```

Useful z/OS UNIX commands

Refer to *UNIX System Services Command Reference* (SA22-7802) for more information on the following sample z/OS UNIX commands:

- Use the following z/OS UNIX **ls** command to display all files within a directory.

  ```
  ls -lR /var/rdz/pushtoclient/
  ```

- Use the following z/OS UNIX **chown** command to change the owner of a directory and all files within.

  ```
  chown –R IBMUSER /var/rdz/pushtoclient/
  ```

- Use the following z/OS UNIX **chgrp** command to assign the group to the directory and all files within.

  ```
  chgrp -R RDZPROJ /var/rdz/pushtoclient/
  ```

- Use the following z/OS UNIX **chmod** command to give the owner and group write permission to the directory and all files within. Other has read permission. All have execute permission.

  ```
  chmod -R 775 /var/rdz/pushtoclient/
  ```

Sample setup

In the following scenario, all the development project managers, a team of three, are tasked with being a Developer for System z client administrator.

The security administrator has already assigned a default group (RDZPROJ) with unique group ID (1200) to the team. Their user IDs do not have special privileges (like UID 0) in z/OS UNIX. The security administrator has not defined the FILE.GROUPOWNER.SETGID profile, so z/OS UNIX will use the group ID of the directory when creating new files. User ID IBMUSER (with UID 0 and default group SYS1) was used by the systems programmer to create directory /var/rdz/pushtoclient.

1. The systems programmer limits /var/rdz/pushtoclient write permissions to the owner and group:

   ```
   # chmod 775 /var/rdz/pushtoclient
   # ls -ld /var/rdz/pushtoclient
   drwxrwxr-x   2 IBMUSER  SYS1
   /var/rdz/pushtoclient
   ```

   Note: The FEKSETUP job used during customization setup already does this step.

2. The systems programmer makes RDZPROJ the owning group:

   ```
   # chgrp RDZPROJ /var/rdz/pushtoclient
   # ls –ld /var/rdz/pushtoclient
   drwxrwxr-x   2 IBMUSER  RDZPROJ
   /var/rdz/pushtoclient
   ```

This concludes the setup required to limit /var/rdz/pushtoclient write permissions to the systems programmer (IBMUSER) and the project managers (RDZPROJ).

Chapter 2. Security considerations

Developer for System z provides mainframe access to users on a non-mainframe workstation. Validating connection requests, providing secure communication between the host and the workstation, and authorizing and auditing activity are therefore important aspects of the product configuration.

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

The following topics are covered in this chapter:

Note: Remote Systems Explorer (RSE), which provides core services such as connecting the client to the host, consists of 2 logical entities:

- RSE daemon, which manages connection setup, and is started as a started task or long running user job.
- RSE server, which handles individual client request, and is started as a thread in one or more child processes by RSE daemon.

Refer to Chapter 1, "Understanding Developer for System z," on page 1 to learn about basic Developer for System z design concepts.

Authentication methods

Developer for System z supports multiple ways to authenticate a user ID provided by a client upon connection.

- User ID and password
- User ID and one-time password
- X.509 certificate

Note: The authentication data provided by the client is only used once, during initial connection setup. After a user ID is authenticated, the user ID and self-generated PassTickets are used for all actions that require authentication.

User ID and password

The client provides a user ID and matching password upon connection. The user ID and password are used to authenticate the user with your security product.

User ID and one-time password

Based upon a unique token, a one-time password can be generated by a third-party product. One-time passwords improve your security setup as the unique token cannot be copied and used without the user's knowledge, and an intercepted password is useless because it is valid only once.

The client provides a user ID and the one-time password upon connection, which is used to authenticate the user ID with the security exit provided by the third party. This security exit is expected to ignore the PassTickets used to satisfy authentication requests during normal processing. The PassTickets must be processed by your security software.

X.509 certificate

A third party can provide one or more X.509 certificates that can be used for authenticating a user. When stored on secure devices, X.509 certificates combine a secure setup with ease of use for the user (no user ID or password needed).

Upon connection, the client provides a selected certificate, and optionally a selected extension, which is used to authenticate the user ID with your security product.

Note: This authentication method is only supported by the RSE daemon connection method, and SSL (Secure Socket Layer) communication must be enabled.

JES Job Monitor authentication

Client authentication is done by RSE daemon (or REXEC/SSH) as part of the client's connection request. Once the user is authenticated, self-generated PassTickets are used for all future authentication requests, including the automatic logon to JES Job Monitor.

In order for JES Job Monitor to validate the user ID and PassTicket presented by RSE, JES Job Monitor must be allowed to evaluate the PassTicket. This implies the following:

- Load module `FEJJMON`, by default located in load library `FEK.SFEKAUTH`, must be APF authorized.
- Both RSE and JES Job Monitor must use the same application ID (APPLID). By default both servers use `FEKAPPL` as APPLID, but this can be changed by the APPLID directive in `rsed.envvars` for RSE and in `FEJJCNFG` for JES Job Monitor.

Note: Previous clients (version 7.0 and older) communicate directly with JES Job Monitor. For these connections, only the user ID and password authentication method is supported.

Connection security

Different levels of communication security are supported by RSE, which controls all communication between the client and Developer for System z services:

- External (client-host) communication can be limited to specified ports. This feature is disabled by default.
- External (client-host) communication can be encrypted using SSL. This feature is disabled by default.
- Port Of Entry (POE) checking can be used to allow host access only to trusted TCP/IP addresses. This feature is disabled by default.

Limit external communication to specified ports

The system programmer can specify the ports on which the RSE server can communicate with the client. By default, any available port is used. This range of ports has no connection with the RSE daemon port.

To help understand the port usage, a brief description of RSE's connection process follows:

1. The client connects to host port 4035, RSE daemon.
2. The RSE daemon creates an RSE server thread.
3. The RSE server opens a host port for the client to connect. The selection of this port can be configured by the user, either on the client in the subsystem properties tab (this is not recommended) or through the `_RSE_PORTRANGE` definition in `rsed.envvars`.
4. The RSE daemon returns the port number to the client.
5. The client connects to the host port.

Note: The process is similar for the (optional) alternative connection method using REXEC/SSH, which is described in "(Optional) Using REXEC (or SSH)" in the *Host Configuration Guide* (SC23-7658).

Communication encryption using SSL

All external Developer for System z data streams that pass through RSE can be encrypted using Secure Socket Layer (SSL). The usage of SSL is controlled by the settings in the `ssl.properties` configuration file, as described in “SSL encrypted communication” on page 23.

The Host Connect Emulator on the client connects to a TN3270 server on the host. The usage of SSL is controlled by TN3270, as documented in the *Communications Server IP Configuration Guide* (SC31-8775).

The Application Deployment Manager client uses the CICS TS Web Service or the RESTful interface to invoke the Application Deployment Manger host services. The usage of SSL is controlled by CICS TS, as documented in *RACF Security Guide for CICS TS*.

Port Of Entry checking

Developer for System z supports Port Of Entry (POE) checking, which allows host access only to trusted TCP/IP addresses. The usage of POE is controlled by the definition of specific profiles in your security software and the `enable.port.of.entry` directive in `rsed.envvars`, as described in “Port Of Entry (POE) checking” on page 27.

Note that activating POE will impact other TCPIP applications that support POE checking, such as INETD.

Using PassTickets

After logon, PassTickets are used to establish thread security within the RSE server. This feature cannot be disabled. PassTickets are system generated passwords with a lifespan of about 10 minutes. The generated PassTickets are based upon the DES encryption algorithm, the user ID, the application ID, a time and date stamp, and a secret key. This secret key is a 64 bit number (16 hex characters) that must be defined to your security software. For additional security, z/OS security software handles PassTickets by default as single-use passwords.

To help understand the PassTicket usage, a brief description of RSE's security process follows:

1. The client connects to host port 4035, RSE daemon.
2. The RSE daemon authenticates the client, using the credentials presented by the client.
3. The RSE daemon creates a unique client ID and an RSE server thread.
4. The RSE server generates a PassTicket and creates a security environment for the client, using the PassTicket as password.
5. The client connects to the host port returned by the RSE daemon.
6. The RSE server validates the client using the client ID.
7. The RSE server uses a newly generated PassTicket as password for all future actions requiring a password.

The actual password of the client is no longer needed after initial authentication because SAF-compliant security products can evaluate both PassTickets and regular passwords. RSE server generates and uses a PassTicket each time a password is required, resulting in a (temporary) valid password for the client.

Using PassTickets allows RSE to set up a user-specific security environment at will, without the need of storing all user IDs and passwords in a table, which could be compromised. It also allows for client authentication methods that do not use reusable passwords, such as X.509 certificates.

Security profiles in the `APPL` and `PTKTDATA` classes are required to be able to use PassTickets. These profiles are application specific and thus do not impact your current system setup.

PassTickets being application specific implies that both RSE and JES Job Monitor must use the same application ID (APPLID). By default both servers use `FEKAPPL` as APPLID, but this can be changed by the `APPLID` directive in `rsed.envvars` for RSE and in `FEJJCNFG` for JES Job Monitor.

You should not use `OMVSAPPL` as application ID, because it will open the secret key to most z/OS UNIX applications. You should also not use the default MVS application ID, which is `MVS` followed by the system's SMF ID, because this will open the secret key to most MVS applications (including user batch jobs).

The smallest unit of a PassTicket timestamp is 1 second. This implies that all PassTickets generated within a single second by the same application for the same user ID will be identical. This, combined with z/OS security software handling PassTickets as single-use passwords, causes a problem for Developer for System z

during logon, as multiple PassTickets will be required within a second. Therefore, Developer for System z requires setting a flag in the PassTicket definitions that allows the generated PassTickets to be reused.

Attention: The client connection request will fail if PassTickets are not set up correctly.

Audit logging

Developer for System z supports audit logging of actions that are managed by the RSE daemon. The audit logs are stored as text files in the daemon log directory, using the CSV (Comma Separated Value) format.

Audit control

Multiple options in `rsed.envvars` influence the audit function, as documented in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC23-7658).

- The audit function is enabled/disabled by the `enable.audit.log` option.
- The audit defaults are controlled by the `audit.*` options.
- The location of the audit log files is controlled by the `daemon.log` option.
- The code page used for writing the audit log is controlled by the `_RSE_HOST_CODEPAGE` directive, as documented in "rsed.envvars, RSE configuration file" in the *Host Configuration Guide* (SC23-7658).

The **modify switch** operator command can be used to manually switch to a new audit log file, as documented in "Operator commands" in the *Host Configuration Guide* (SC23-7658).

A warning message is sent to the console when the file system holding the audit log files is running low on free space. This console message (FEK103E) is repeated regularly until the low space issue is resolved. Refer to "Console messages" in the *Host Configuration Guide* (SC23-7658) for a list of console messages generated by RSE.

Audit processing

A new audit log file is started after a predetermined time or when the **modify switch** operator command is issued. The old log file is saved as `audit.log.yyyymmdd.hhmmss`, where `yyyymmdd.hhmmss` is the date/timestamp when this log was closed. The system date/timestamp assigned to the file indicates the creation of the log file. The combination of the two dates shows the time period covered by this audit log file.

The `audit.action*` directives in `rsed.envvars` allow you to specify a user exit (z/OS UNIX shell script, z/OS UNIX REXX, or z/OS UNIX program) which will be invoked by RSE when an audit log is closed. This user exit can then process the data within the audit log.

Audit log files have permission bit mask 640 (-rw-r-----), if not changed by the `audit.log.mode` directive in `rsed.envvars`. This means that the owner (RSE daemon z/OS UNIX uid) has read and write access, and the owner's (default) group has read access. All other access attempts are denied, unless it is done by a super user (UID 0) or somebody with sufficient permission to the `SUPERUSER.FILESYS` profile in the `UNIXPRIV` security class.

Audit data

The following actions are logged:

- System access (connect, disconnect)
- JES spool access (submit, display, hold, release, cancel, purge)
- Data set access (read, write, create, delete, rename, compress, migration, recall)
- Execution of TSO commands

Each logged action is stored (with a date/timestamp) using the CSV (Comma Separated Value) format, which can be read by an automation or data analysis tool.

JES security

Developer for System z allows clients access to the JES spool through the JES Job Monitor. The server provides basic access limitations, which can be extended with the standard spool file protection features of your security product. Operator actions (Hold, Release, Cancel, and Purge) against spool files are done through an EMCS console, for which conditional permits must be set up.

Actions against jobs - target limitations

JES Job Monitor does not provide Developer for System z users full operator access to the JES spool. Only the Hold, Release, Cancel, and Purge commands are available, and by default, only for spool files owned by the user. The commands are issued by selecting the appropriate option in the client menu structure (there is no command prompt). The scope of the commands can be widened, using security profiles to define for which jobs the commands are available.

Similar to the SDSF **SJ** action character, JES Job Monitor also supports the Show JCL command to retrieve the JCL that created the selected job output, and show it in an editor window. JES Job Monitor retrieves the JCL from JES, making it a useful function for situations in which the original JCL member is not easily located.

Table 1. JES Job Monitor console commands

Action	JES2	JES3
Hold	`$Hx(jobid)` with x = {J, S or T}	`*F,J=jobid,H`
Release	`$Ax(jobid)` with x = {J, S or T}	`*F,J=jobid,R`
Cancel	`$Cx(jobid)` with x = {J, S or T}	`*F,J=jobid,C`
Purge	`$Cx(jobid),P` with x = {J, S or T}	`*F,J=jobid,C`
Show JCL	`not applicable`	`not applicable`

The available JES commands listed in Table 1 are by default limited to jobs owned by the user. This can be changed with the `LIMIT_COMMANDS` directive, as documented in "FEJJCNFG, JES Job Monitor configuration file" in the *Host Configuration Guide* (SC23-7658).

Table 2. LIMIT_COMMANDS command permission matrix

	Job owner	
LIMIT_COMMANDS	**User**	**Other**
USERID (default)	Allowed	Not allowed
LIMITED	Allowed	Allowed only if explicitly permitted by security profiles
NOLIMIT	Allowed	Allowed if permitted by security profiles or when the JESSPOOL class is not active

JES uses the JESSPOOL class to protect SYSIN/SYSOUT data sets. Similar to SDSF, JES Job Monitor extends the use of the JESSPOOL class to protect job resources as well.

If LIMIT_COMMANDS is not USERID, then JES Job Monitor will query for permission to the related profile in the JESSPOOL class, as shown in the following table.

Table 3. Extended JESSPOOL profiles

Command	JESSPOOL profile	Required access
Hold	nodeid.userid.jobname.jobid	ALTER
Release	nodeid.userid.jobname.jobid	ALTER
Cancel	nodeid.userid.jobname.jobid	ALTER
Purge	nodeid.userid.jobname.jobid	ALTER
Show JCL	nodeid.userid.jobname.jobid.JCL	READ

Use the following substitutions in the preceding table:

nodeid	NJE node ID of the target JES subsystem
userid	Local user ID of the job owner
jobname	Name of the job
jobid	JES job ID

If the JESSPOOL class is not active, then there is different behavior for the LIMITED and NOLIMIT value of LIMIT_COMMANDS, as described in the "LIMIT_COMMANDS command permission matrix table" in "FEJJCNFG, JES Job Monitor Configuration file" in the *Host Configuration Guide* (SC23-7658). The behavior is identical when JESSPOOL is active, since the class, by default, denies permission if a profile is not defined.

Actions against jobs - execution limitations

The second phase of JES spool command security, after specifying the permitted targets, includes the permits needed to actually execute the operator command. This execution authorization is enforced by the z/OS and JES security checks.

Note that Show JCL is not an operator command such as the other JES Job Monitor commands (Hold, Release, Cancel, and Purge), so the limitations in the next list do not apply because there is no further security check.

JES Job Monitor issues all JES operator commands requested by a user through an extended MCS (EMCS) console, whose name is controlled with the CONSOLE_NAME directive, as documented in "FEJJCNFG, JES Job Monitor configuration file" in the *Host Configuration Guide* (SC23-7658).

This setup allows the security administrator to define granular command execution permits using the OPERCMDS and CONSOLE classes.

- In order to use an EMCS console, a user must have (at least) READ authority to the MVS.MCSOPER.console-name profile in the OPERCMDS class. Note that if no profile is defined, the system will grant the authority request.
- In order to execute a JES operator command, a user must have sufficient authority to the JES%.** (or more specific) profile in the OPERCMDS class. Note that if no profile is defined, or the OPERCMDS class is not active, JES will fail the command.
- The security administrator can also require that a user must use JES Job Monitor when executing the operator command by specifying WHEN(CONSOLE(JMON)) on the **PERMIT** definition. The CONSOLE class must be active for this setup to work. Note that the CONSOLE class being active is sufficient; no profiles are checked for EMCS consoles.

Assuming the identity of the JES Job Monitor server by creating a JMON console from a TSO session is prevented by your security software. Even though the console can be created, the point of entry is different (JES Job Monitor versus TSO). JES commands issued from this console will fail the security check, if your security is set up as documented in this publication and the user does not have authority to JES commands through other means.

Note that JES Job Monitor cannot create the console when a command must be executed if the console name is already in use. To prevent this, the system programmer can set the GEN_CONSOLE_NAME=ON directive in the JES Job Monitor configuration file or the security administrator can define security profiles to stop TSO users from creating a console. The following sample RACF commands prevent everyone (except those permitted) from creating a TSO or SDSF console:

- RDEFINE TSOAUTH CONSOLE UACC(NONE)
- PERMIT CONSOLE CLASS(TSOAUTH) ACCESS(READ) ID(#userid)
- RDEFINE SDSF ISFCMD.ODSP.ULOG.* UACC(NONE)
- PERMIT ISFCMD.ODSP.ULOG.* CLASS(SDSF) ACCESS(READ) ID(#userid)

Note: Without being authorized for these operator commands, users will still be able to submit jobs and read job output through the JES Job Monitor, if they have sufficient authority to possible profiles that protect these resources (such as those in the JESINPUT, JESJOBS and JESSPOOL classes).

Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information on operator command protection.

Access to spool files

JES Job Monitor allows browse access to all spool files by default. This can be changed with the LIMIT_VIEW directive, as documented in "FEJJCNFG, JES Job Monitor configuration file" in the *Host Configuration Guide* (SC23-7658).

Table 4. LIMIT_VIEW browse permission matrix

	Job owner	
LIMIT_VIEW	**User**	**Other**
USERID	Allowed	Not allowed
NOLIMIT (default)	Allowed	Allowed if permitted by security profiles or when the JESSPOOL class is not active

To limit users to their own jobs on the JES spool, define the "LIMIT_VIEW=USERID" statement in the JES Job Monitor configuration file, FEJJCNFG. If the users need access to a wider range of jobs, but not all, use the standard spool file protection features of your security product, such as the JESSPOOL class.

When defining further protection, keep in mind that JES Job Monitor uses SAPI (SYSOUT application program interface) to access the spool. This implies that the user needs at least UPDATE access to the spool files, even for browse functionality. This requisite does not apply if you run z/OS 1.7 (z/OS 1.8 for JES3) or higher. Here READ permission is sufficient for browse functionality.

Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information on JES spool file protection.

SSL encrypted communication

External (client-host) communication can be encrypted using SSL (Secure Socket Layer). This feature is disabled by default and is controlled by the settings in ssl.properties. Refer to "(Optional) ssl.properties, RSE SSL encryption" in the *Host Configuration Guide* (SC23-7658).

RSE daemon and RSE server support different mechanisms to store certificates due to architectural differences between the two. This implies that SSL definitions and certificates are required for both RSE daemon and RSE server. A shared certificate can be used if RSE daemon and RSE server use the same certificate management method.

Table 5. SSL certificate storage mechanisms

Certificate storage	Created and managed by	RSE daemon	RSE server
key ring	SAF compliant security product	supported	supported
key database	z/OS UNIX's gskkyman	supported	/
key store	Java's keytool	/	supported

Note: SAF-compliant key rings are the preferred method for managing certificates.

SAF-compliant key rings can store the certificate's private key either in the security database or by using ICSF (Integrated Cryptographic Service Facility), the interface to System z cryptographic hardware.

ICSF is recommended for the storage of the private keys associated with digital certificates, because it is a more secure solution than non-ICSF private key

management. ICSF ensures that private keys are encrypted under the ICSF master key and that access to them is controlled by general resources in the CSFKEYS and CSFSERV security classes. In addition, operational performance is improved because ICSF utilizes the hardware Cryptographic Coprocessor. See *Cryptographic Services ICSF Administrator's Guide* (SA22-7521) for more details about ICSF and how to control who can use cryptographic keys and services.

RSE daemon uses System SSL functions to manage SSL encrypted communications. This implies that SYS1.SIEALNKE must be program controlled by your security software and available to RSE via LINKLIST or the STEPLIB directive in rsed.envvars.

The RSE user ID (stcrse in the following sample commands) needs authorization to access his key ring and the related certificates when SAF-compliant key rings are used for either RSE daemon or RSE server.

- RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
- RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
- PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ACCESS(READ) ID(stcrse)
- PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ) ID(stcrse)
- SETROPTS RACLIST(FACILITY) REFRESH

Refer to Appendix A, "Setting up SSL and X.509 authentication," on page 155 for more details on activating SSL for Developer for System z.

Client authentication using X.509 certificates

RSE daemon supports users authenticating themselves with an X.509 certificate. Using SSL encrypted communication is a prerequisite for this function, as it is an extension to the host authentication with a certificate used in SSL.

RSE daemon starts the client authentication process by validating the client certificate. Some key aspects that are checked are the dates the certificate is valid and the trust-worthiness of the Certificate Authority (CA) used to sign the certificate. Optionally, a (third party) Certificate Revocation List (CRL) can also be consulted.

After RSE daemon validates the certificate, it is processed for authentication. The certificate is passed on to your security product for authentication, unless rsed.envvars directive enable.certificate.mapping is set to false, at which point RSE daemon will do the authentication.

If successful, the authentication process will determine the user ID to be used for this session, which is then tested by RSE daemon to ensure it is usable on the host system where RSE daemon is running.

The last check (which is done for every authentication mechanism, not just X.509 certificates) verifies that the user ID is allowed to use Developer for System z.

If you are familiar with the SSL security classifications used by TCP/IP, the combination of these validation steps match the "Level 3 Client authentication" specifications (the highest available).

Certificate Authority (CA) validation

Part of the certificate validation process includes checking that the certificate was signed by a Certificate Authority (CA) you trust. In order to do so, RSE daemon must have access to a certificate that identifies the CA.

When using the **gskkyman** key database for your SSL connection, the CA certificate must be added to the key database.

When using an SAF key ring (which is the advised method), you must add the CA certificate to your security database as a CERTAUTH certificate with the TRUST or HIGHTRUST attribute, as shown in this sample RACF command:

- `RACDCERT CERTAUTH ADD(dsn) HIGHTRUST WITHLABEL('label')`

Note that most security products already have the certificates for well known CA's available in their database with a NOTRUST status. Use the following sample RACF commands to list the existing CA certificates and mark one as trusted based on the label assigned to it.

- `RACDCERT CERTAUTH LIST`
- `RACDCERT CERTAUTH ALTER(LABEL('HighTrust CA')) HIGHTRUST`

Note: The HIGHTRUST status is required if you rely on RACF authenticating the user based upon the HostIdMappings extension in the certificate. Refer to "Authentication by your security software" on page 26 for more information.

Once the CA certificate is added to your security database, it must be connected to the RSE key ring, as shown in this sample RACF command:

- `RACDCERT ID(stcrse) CONNECT(CERTAUTH LABEL('HighTrust CA')`
  `RING(rdzssl.racf))`

Refer to *Security Server RACF Command Language Reference* (SA22-7687) for more information on the **RACDCERT** command.

Attention: If you rely on RSE daemon instead of your security software to authenticate a user you must be cautious not to mix CAs with a TRUST and HIGHTRUST status in your `SAF` key ring or **gskkyman** key database. RSE daemon is not able to differentiate between the two, so certificates signed by a CA with TRUST status will be valid for user ID authentication purposes.

(Optional) Query a Certificate Revocation List (CRL)

If desired, you can instruct RSE daemon to check one or more Certificate Revocation List(s) (CRL) to add extra security to the validation process. This is done by adding CRL-related environment variables to `rsed.envvars`.

- GSK_CRL_SECURITY_LEVEL
- GSK_LDAP_SERVER
- GSK_LDAP_PORT
- GSK_LDAP_USER
- GSK_LDAP_PASSWORD

Refer to the *Cryptographic Services System Secure Sockets Layer Programming* (SC24-5901) for more information on these and other environment variables used by z/OS System SSL.

Note: Be careful when specifying other z/OS System SSL environment variables (`GSK_*`) in `rsed.envvars`, as they might change the way RSE daemon handles SSL connections and certificate authentication.

Authentication by your security software

RACF performs several checks to authenticate a certificate and return the associated user ID. Note that other security products might do this differently. Refer to your security product documentation for more information on the `initACEE` function used to do the authentication (query mode).

1. RACF checks if the certificate is defined in the `DIGTCERT` class. If so, RACF returns the user ID that was associated with this certificate when it was added to the RACF database.

   Certificates are defined to RACF using the `RACDCERT` command, as in the following example:

   ```
   RACDCERT ID(userid) ADD(dsn) TRUST WITHLABEL('label')
   ```

2. If the certificate is not defined, RACF checks to see if there is a matching certificate name filter defined in the `DIGTNMAP` or `DIGTCRIT` classes. If so, it returns the user ID associated with the most specific matching filter.

   Note: It is advised not to use name filters for certificates used by Developer for System z, as these filters map all certificates to a single user ID. The result is that all your Developer for System z users will log on with the same user ID.

3. If there is no matching name filter, RACF locates the HostIdMappings certificate extension and extracts the embedded user ID and host name pair. If found and validated, RACF returns the user ID defined within the HostIdMappings extension.

   The user ID and host name pair is valid if all these conditions are true:

   - The CA certificate used to sign this certificate is marked as HIGHTRUST in the `DIGTCERT` class.
   - The user ID stored in the extension has a valid length (1 to 8 characters).
   - The user ID assigned to RSE daemon has (at least) READ authority to the `IRR.HOST.hostname` profile in the `SERVAUTH` class, where `hostname` is the host name stored in the extension. This is usually a domain name, such as CDFMVS08.RALEIGH.IBM.COM.

   The definition of the HostIdMappings extension in ASN.1 syntax is:

   ```
   id-ce-hostIdMappings OBJECT IDENTIFIER::= {1 3 18 0 2 18 1}
   HostIdMappings::= SET OF HostIdMapping
   HostIdMapping::= SEQUENCE{
      hostName        IMPLICIT[1] IA5String,
      subjectId       IMPLICIT[2] IA5String,
      proofOfIdPossession IdProof OPTIONAL
    }
    IdProof::= SEQUENCE{
      secret        OCTET STRING,
      encryptionAlgorithm OBJECT IDENTIFIER
    }
   ```

   Note: A HostIdMappings extension is not honored if the target user ID was created after the start of the validity period for the certificate containing the HostIdMappings extension. Therefore, if you are creating user IDs specifically for certificates with HostIdMappings extensions, make sure that you create the user IDs before the certificate requests are submitted.

   Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information on X.509 certificates, how they are managed by RACF, and

how to define certificate name filters. Refer to *Security Server RACF Command Language Reference* (SA22-7687) for more information on the **RACDCERT** command.

Authentication by RSE daemon

Developer for System z can do basic X.509 certificate authentication without relying on your security product. Authentication done by RSE daemon requires a user ID and host name to be defined in a certificate extension, and is only activated if the `enable.certificate.mapping` directive in `rsed.envvars` is set to `FALSE`.

This function is intended to be used if your security product does not support authenticating a user based upon an X.509 certificate, or if your certificate would fail the test(s) done by your security product (for example, the certificate has a faulty identifier for the HostIdMappings extension and there is no name filter or definition in `DIGTCERT`).

The client will query the user for the extension identifier (OID) to use, which is by default the HostIdMappings OID, {1 3 18 0 2 18 1}.

RSE daemon will extract the user ID and host name from it using the format of the HostIdMappings extension. This format is described in "Authentication by your security software" on page 26.

The user ID and host name pair is valid if all these conditions are true:

- The user ID stored in the extension has a valid length (1 to 8 characters).
- The user ID assigned to RSE daemon has (at least) READ authority to the `IRR.HOST.hostname` profile in the `SERVAUTH` class, where `hostname` is the host name stored in the extension. This is usually a domain name, such as CDFMVS08.RALEIGH.IBM.COM.

Attention: It is up to the security administrator to ensure that all CAs known to RSE daemon are highly trusted, because RSE daemon cannot check if the one who signed the client certificate is highly trusted or just trusted. See "Certificate Authority (CA) validation" on page 25 for more information on accessible CA certificates.

Port Of Entry (POE) checking

Developer for System z supports Port Of Entry (POE) checking, which allows host access only to trusted TCP/IP addresses. This feature is disabled by default and requires the definition of the `BPX.POE` security profile, as shown in the following sample RACF commands:

- `RDEFINE FACILITY BPX.POE UACC(NONE)`
- `PERMIT BPX.POE CLASS(FACILITY) ACCESS(READ) ID(STCRSE)`
- `SETROPTS RACLIST(FACILITY) REFRESH`

Note:

- RSE must be configured to use POE by uncommenting the "`enable.port.of.entry=true`" option in `rsed.envvars`, as documented in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC23-7658).

- The RSE user ID STCRSE requires UID(0) when this profile is not defined and POE checking is enabled in rsed.envvars.
- Defining BPX.POE will impact other TC/PIP applications that support POE checking, such as INETD.
- Security zones (EZB.NETACCESS.** profiles, which are IP address ranges) should be set up in the SERVAUTH class to use the full strength of POE checking.

Refer to *Communications Server IP Configuration Guide* (SC31-8775) for more information on network access control using POE checking.

Managed ACEE

Developer for System z uses z/OS UNIX kernel services, such as pthread_security_np() and __passwd(), that use the InitACEE security service, resulting in "managed ACEE" security control blocks. A managed ACEE (Accessor Environment Element) is cached by your security product, and your security product will ignore certain changes, (such as password changes outside of Developer for System z) until the cache times out. (Timing out can take a few minutes.)

Refresh the managed ACEE cache after security changes to ensure that the new data is used by Developer for System z.

Push-to-client developer groups

Developer for System z clients version 8.0.1 and higher can pull client configuration files and upgrade information from the host when they connect, ensuring that all clients have common settings and that they are up-to-date.

Since version 8.0.3, 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.

When using definitions in your security database as selection mechanism (the SAF value is specified for directives in pushtoclient.properties), Developer for System z verifies access permits to the profiles listed in Table 6 to determine which developer groups the user belongs to, and whether a user is allowed to reject updates.

Table 6. Push-to-client SAF information

FACILITY profile	Fixed length	Required access	Result
FEK.PTC.CONFIG.ENABLED. sysname.devgroup	23	READ	Client accepts configuration updates for the specified group
FEK.PTC.PRODUCT. ENABLED.sysname.devgroup	24	READ	Client accepts product updates for the specified group
FEK.PTC.REJECT.CONFIG. UPDATES.sysname	30	READ	User can reject configuration updates

Table 6. Push-to-client SAF information (continued)

FACILITY profile	Fixed length	Required access	Result
FEK.PTC.REJECT.PRODUCT. UPDATES.sysname	31	READ	User can reject product updates

Note: Developer for System z assumes that a user has no access authorization when your security software indicates it cannot determine whether or not a user has access authorization to a profile. An example of this is when the profile is not defined.

The `devgroup` value matches the group name assigned to a specific group of developers. Note that the group name is visible on Developer for System z clients.

The `sysname` value matches the system name of the target system.

The "Fixed length" column documents the length of the fixed part of the related security profile.

By default, Developer for System z expects the `FEK.PTC.*` profiles to be in the `FACILITY` security class. Note that profiles in the FACILITY class are limited to 39 characters. If the sum of the length of the fixed profile part (FEK.PTC.<key>) and the length of the site-specific profile part (`sysname` or `sysname.devgroup`) exceeds this number you can place the profiles in another class and instruct Developer for System z to use this class instead. To do so, uncomment `_RSE_FEK_SAF_CLASS` in `rsed.envvars` and provide the desired class name.

Note that the client administrator must be on the access list of the `FEK.PTC.*.ENABLED.*` profiles to define and manage the related push-to-client metadata. This implies that the profiles must be defined with (at least) the client administrator on the access list before push-to-client with group support can be implemented.

See "(Optional) pushtoclient.properties, Host-based client control" in the *Host Configuration Guide* (SC23-7658) for more information about enabling multiple group support. See Chapter 7, "Push-to-client considerations," on page 99 for more information about push-to-client concepts and implementation.

CICSTS security

Developer for System z allows, through Application Deployment Manager, CICS administrators to control which CICS resource definitions are editable by the developer, their default values, and the display of a CICS resource definition by means of the CICS Resource Definition (CRD) server. Refer to Chapter 8, "CICSTS considerations," on page 115 for more information on the required CICS TS security definitions.

CRD repository

The CRD server repository VSAM data set holds all the default resource definitions and must therefore be protected against updates, but developers must be allowed to read the values stored here.

CICS transactions

Developer for System z supplies multiple transactions that are used by the CRD server when defining and inquiring CICS resources. When the transaction is attached, CICS resource security checking, if enabled, insures that the user ID is authorized to run the transaction ID.

SSL encrypted communication

The Application Deployment Manager client uses CICS TS Web Services or the RESTful interface to invoke the CRD server. The usage of SSL for this communication is controlled by the CICS TS `TCPIPSERVICE` definition, as documented in *RACF Security Guide for CICS TS*.

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.

Developer for System z configuration files

There are several Developer for System z configuration files whose directives impact the security and audit setup. Based upon the information in this chapter, the security administrator and systems programmer can decide what the settings should be for the following directives.

JES Job Monitor - FEJJCNFG

- `LIMIT_COMMANDS={USERID | LIMITED | NOLIMIT}`

  Define against which jobs actions can be done (excluding browse and submit). For more information, see "Actions against jobs - target limitations" on page 20.
- `LIMIT_VIEW={USERID | NOLIMIT}`

  Define which spool files can be browsed. For more information, see "Access to spool files" on page 22.
- `APPLID={FEKAPPL | *}`

  Application ID used for PassTicket creation/validation. For more information, see "Using PassTickets" on page 18.

Note: Details on these and other `FEJJCNFG` directives are available in "FEJJCNFG, JES Job Monitor configuration file" in the *Host Configuration Guide* (SC23-7658).

RSE - rsed.envvars

- `_RSE_FEK_SAF_CLASS={FACILITY | *}`

  Security class holding `FEK.PTC.**` profiles. For more information, see "Push-to-client developer groups" on page 28.
- `(_RSE_JAVAOPTS) -DDENY_PASSWORD_SAVE={true | false}`

Deny users to save their host password on the client. For more information, see "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC23-7658).

- `(_RSE_JAVAOPTS) -DDSTORE_IDLE_SHUTDOWN_TIMEOUT=value`

  Timer to disconnect idle clients. For more information, see "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC23-7658).

- `(_RSE_JAVAOPTS) -DAPPLID={FEKAPPL | *}`

  Application ID used for PassTicket creation/validation. For more information, see “Using PassTickets” on page 18.

- `(_RSE_JAVAOPTS) -Denable.port.of.entry={true | false}`

  Enable Port Of Entry checking. For more information, see “Port Of Entry (POE) checking” on page 27.

- `(_RSE_JAVAOPTS) -Denable.certificate.mapping={true | false}`

  Use your security product to authenticate users with an X.509 certificate. For more information, see “Client authentication using X.509 certificates” on page 24.

- `GSK_CRL_SECURITY_LEVEL={LOW | MEDIUM | HIGH}`

  ```
  GSK_LDAP_SERVER=*
  GSK_LDAP_PORT={389 | *}
  GSK_LDAP_USER=*
  GSK_LDAP_PASSWORD=*
  ```

  Additional security checks for X.509 authentication. For more information, see “(Optional) Query a Certificate Revocation List (CRL)” on page 25.

- `(_RSE_JAVAOPTS) -Ddaemon.log={/var/rdz/logs | *}`

  Location of the audit log files. For more information, see “Audit logging” on page 19.

- `(_RSE_JAVAOPTS) -Daudit.log.mode={RW.R. | * }`

  File access permission mask of the audit log files. For more information, see “Audit logging” on page 19.

- `(_RSE_JAVAOPTS) -Daudit.action=<shell script>`

  `(_RSE_JAVAOPTS) -Daudit.action.id=<userid>`

  z/OS UNIX based user exit that processes audit logs. For more information, see “Audit logging” on page 19.

Note: Details on these and other `rsed.envvars` directives are available in "rsed.envvars, RSE configuration file" in the *Host Configuration Guide* (SC23-7658).

RSE - ssl.properties

- `daemon_keydb_file={SAF key ring name | gskkyman key database name}`

  Location of the RSE daemon certificate. For more information, see “SSL encrypted communication” on page 23.

- `daemon_key_label=certificate label`

  Name of the RSE daemon certificate. For more information, see “SSL encrypted communication” on page 23.

- `server_keystore_file={SAF key ring name | Java key store name}`

  Location of the RSE server certificate. For more information, see “SSL encrypted communication” on page 23.

- `server_keystore_label=certificate label`

  Name of the RSE server certificate. For more information, see “SSL encrypted communication” on page 23.

- `server_keystore_type={JKS | JCERACFKS | JCECCARACFKS}`

  Type of key store used (Java key store or SAF key ring). For more information, see “SSL encrypted communication” on page 23.

Note: Details on these and other `ssl.properties` directives are available in "(Optional) ssl.properties, RSE SSL encryption" in the *Host Configuration Guide* (SC23-7658).

RSE - pushtoclient.properties

- `config.enabled={true | false | SAF | LDAP}`
  `reject.config.updates={true | false | SAF | LDAP}`

  Host-based control of Developer for System z client configuration files. For more information, see Chapter 7, “Push-to-client considerations,” on page 99.
- `product.enabled={true | false | SAF | LDAP}`
  `reject.product.updates={true | false | SAF | LDAP}`

  Host-based control of Developer for System z client product updates. For more information, see Chapter 7, “Push-to-client considerations,” on page 99.

Note: Details on these and other `pushtoclient.properties` directives are available in "(Optional) pushtoclient.properties, Host-based client control" in the *Host Configuration Guide* (SC23-7658).

Security definitions

Customize and submit sample member `FEKRACF`, which has sample RACF and z/OS UNIX commands to create the basic security definitions for Developer for System z.

`FEKRACF` is located in `FEK.#CUST.JCL`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See "Customization setup" in the *IBM Rational Developer for System z Host Configuration Guide* for more details.

Refer to the *RACF Command Language Reference* (SA22–7687), for more information about RACF commands.

Note:

- For those sites that use CA ACF2™ for z/OS, please refer to your product page on the CA support site (`https://support.ca.com`) and check for the related Developer for System z Knowledge Document, `TEC492389`. This Knowledge Document has details on the security commands necessary to properly configure Developer for System z.
- For those sites that use CA Top Secret® for z/OS, please refer to your product page on the CA support site (`https://support.ca.com`) and check for the related Developer for System z Knowledge Document, `TEC492091`. This Knowledge Document has details on the security commands necessary to properly configure Developer for System z.

The following sections describe the required steps, optional configuration and possible alternatives.

Requirements and checklist

To complete the security setup, the security administrator needs to know the values listed in Table 7. These values were defined during previous steps of the installation and customization of Developer for System z.

Table 7. Security setup variables

Description	• **Default value** • **Where to find the answer**	Value
Developer for System z product high level qualifier	• `FEK` • SMP/E installation	
Developer for System z customization high level qualifier	• `FEK.#CUST` • `FEK.SFEKSAMP(FEKSETUP)`, as described in "Customization setup" in the *IBM Rational Developer for System z Host Configuration Guide*.	
JES Job Monitor started task name	• `JMON` • `FEK.#CUST.PROCLIB(JMON)`, as described in "PROCLIB changes" in the *IBM Rational Developer for System z Host Configuration Guide*	
RSE daemon started task name	• `RSED` • `FEK.#CUST.PROCLIB(RSED)`, as described in "PROCLIB changes" in the *IBM Rational Developer for System z Host Configuration Guide*.	
Lock daemon started task name	• `LOCKD` • `FEK.#CUST.PROCLIB(LOCKD)`, as described in "PROCLIB changes" in the *IBM Rational Developer for System z Host Configuration Guide*.	
Application ID	• `FEKAPPL` • `/etc/rdz/rsed.envvars`, as described in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *IBM Rational Developer for System z Host Configuration Guide*	

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

desired security level. Refer to the previous sections for information about the security setup of optional Developer for System z services.

Activate security settings and classes

Developer for System z utilizes a variety of security mechanisms to ensure a secure and controlled host environment for the client. In order to do so, several classes and security settings must be active, as shown with the following sample RACF commands:

- Display current settings
  -
    ```
    SETROPTS LIST
    ```
- Activate facility class for z/OS UNIX and digital certificate profiles
  -
    ```
    SETROPTS GENERIC(FACILITY)
    ```
  -
    ```
    SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
    ```
- Activate started task definitions
  -
    ```
    SETROPTS GENERIC(STARTED)
    ```
  -
    ```
    RDEFINE STARTED ** STDATA(USER(=MEMBER) GROUP(STCGROUP) TRACE(YES))
    ```
  -
    ```
    SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
    ```
- Activate console security for JES Job Monitor
  -
    ```
    SETROPTS GENERIC(CONSOLE)
    ```
  -
    ```
    SETROPTS CLASSACT(CONSOLE) RACLIST(CONSOLE)
    ```
- Activate operator command protection for JES Job Monitor
  -
    ```
    SETROPTS GENERIC(OPERCMDS)
    ```
  -
    ```
    SETROPTS CLASSACT(OPERCMDS) RACLIST(OPERCMDS)
    ```
- Activate application protection for RSE
  -
    ```
    SETROPTS GENERIC(APPL)
    ```

–

```
SETROPTS CLASSACT(APPL) RACLIST(APPL)
```

- Activate secured signon using PassTickets for RSE
  –

  ```
  SETROPTS GENERIC(PTKTDATA)
  ```

  –

  ```
  SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
  ```

- Activate program control to ensure that only trusted code can be loaded by RSE
  –

  ```
  RDEFINE PROGRAM ** ADDMEM('SYS1.CMDLIB'//NOPADCHK) UACC(READ)
  ```

  –

  ```
  SETROPTS WHEN(PROGRAM)
  ```

  Note: Do not create the ** profile if you already have a * profile in the PROGRAM class. It obscures and complicates the search path used by your security software. In this case, you must merge the existing * and the new ** definitions. IBM recommends to use the ** profile, as documented in *Security Server RACF Security Administrator's Guide* (SA22-7683).

Attention: Some products, such as FTP, require being program controlled if "WHEN PROGRAM" is active. Test this before activating it on a production system.

- (Optional) Activate X.509 HostIdMappings and extended Port Of Entry (POE) support
  –

  ```
  SETROPTS GENERIC(SERVAUTH)
  ```

  –

  ```
  SETROPTS CLASSACT(SERVAUTH) RACLIST(SERVAUTH)
  ```

Define an OMVS segment for Developer for System z users

A RACF OMVS segment (or equivalent) that specifies a valid non-zero z/OS UNIX user ID (UID), home directory, and shell command must be defined for each user of Developer for System z. Their default group also requires an OMVS segment with a group id.

Replace in the following sample RACF commands the #userid, #user-identifier, #group-name and #group-identifier placeholders with actual values:

•

```
ALTUSER #userid
OMVS(UID(#user-identifier) HOME(/u/#userid) PROGRAM(/bin/sh) NOASSIZEMAX)
```

•

```
ALTGROUP #group-name OMVS(GID(#group-identifier))
```

Although it is advised not to do so, you can use the shared OMVS segment defined in the BPX.DEFAULT.USER profile of the FACILITY class to fulfill the OMVS segment requirement for Developer for System z.

Define data set profiles

READ access for users and ALTER for system programmers suffices for most Developer for System z 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. FEK is the default high-level qualifier used during installation and FEK.#CUST is the default high-level qualifier for data sets created during the customization process.

•
```
ADDGROUP (FEK) OWNER(IBMUSER) SUPGROUP(SYS1)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - HLQ STUB')
```
•
```
ADDSD  'FEK.*.**' UACC(READ)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
```
•
```
PERMIT 'FEK.*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
```
•
```
SETROPTS GENERIC(DATASET) REFRESH
```

Note:

- You are strongly advised to protect FEK.SFEKAUTH against updates since this data set is APF authorized. The same is true for FEK.SFEKLOAD and FEK.SFEKLPA, but here because these data sets are program controlled.
- The sample commands in this publication and in the FEKRACF job assume that EGN (Enhanced Generic Naming) is active. This allows the usage of the ** qualifier to represent any number of qualifiers in the DATASET class. Substitute ** with * if EGN is not active on your system. Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information about EGN.

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

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

  –
  ```
  ADDSD  'FEK.#CUST.LSTRANS.*.**' UACC(UPDATE)
   DATA('RATIONAL DEVELOPER FOR SYSTEM Z - SCLMDT')
  ```
  –
  ```
  PERMIT 'FEK.#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, FEK.#CUST.CRA*.

  –
  ```
  ADDSD  'FEK.#CUST.CRA*.**' UACC(READ)
  DATA('RATIONAL DEVELOPER FOR SYSTEM Z - CARMA')
  ```
  –
  ```
  PERMIT 'FEK.#CUST.CRA*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
  ```
  –
  ```
  PERMIT 'FEK.#CUST.CRA*.**' CLASS(DATASET) ACCESS(UPDATE) ID(#ram-developer)
  ```
  –
  ```
  SETROPTS GENERIC(DATASET) REFRESH
  ```
- If Application Deployment Manager's CRD server (CICS Resource Definition) is used, then CICS administrators require UPDATE access to the CRD repository VSAM.

  –

```
ADDSD 'FEK.#CUST.ADNREP*.**' UACC(READ)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - ADN')
-
PERMIT 'FEK.#CUST.ADNREP*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.#CUST.ADNREP*.**' CLASS(DATASET) ACCESS(UPDATE) ID(#cicsadmin)
-
SETROPTS GENERIC(DATASET) REFRESH
```

- If Application Deployment Manager's manifest repository is defined, then all CICS Transaction Server users require UPDATE access to the manifest repository VSAM.

```
-
ADDSD 'FEK.#CUST.ADNMAN*.**' UACC(UPDATE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - ADN')
-
PERMIT 'FEK.#CUST.ADNMAN*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
SETROPTS GENERIC(DATASET) REFRESH
```

Use the following sample RACF commands for a more secure setup where READ access is also controlled.

- uacc(none) data set protection

```
-
ADDGROUP (FEK)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - HLQ STUB')
OWNER(IBMUSER) SUPGROUP(SYS1)"
-
ADDSD 'FEK.*.**'     UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
-
ADDSD 'FEK.SFEKAUTH' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
-
ADDSD 'FEK.SFEKLOAD' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
-
ADDSD 'FEK.SFEKPROC' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
-
ADDSD 'FEK.#CUST.PARMLIB' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
-
ADDSD 'FEK.#CUST.CNTL' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
-
ADDSD 'FEK.#CUST.LSTRANS.*.**' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - SCLMDT')
-
ADDSD 'FEK.#CUST.CRA*.**' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - CARMA')
-
```

```
ADDSD 'FEK.#CUST.ADNREP*.**' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - ADN')
-
ADDSD 'FEK.#CUST.ADNMAN*.**' UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z - ADN')
```

- permit system programmer to manage all libraries

```
-
PERMIT 'FEK.*.**     CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.SFEKAUTH CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.SFEKLOAD CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.SFEKLOAD CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.SFEKLOAD CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.SFEKPROC CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.#CUST.PARMLIB CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.#CUST.CNTL CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.#CUST.LSTRANS.*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.#CUST.CRA*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.#CUST.ADNREP*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
-
PERMIT 'FEK.#CUST.ADNMAN*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
```

- permit clients to access the load and exec libraries

```
-
PERMIT 'FEK.SFEKAUTH' CLASS(DATASET) ACCESS(READ) ID(*)
-
PERMIT 'FEK.SFEKLOAD' CLASS(DATASET) ACCESS(READ) ID(*)
-
PERMIT 'FEK.SFEKPROC' CLASS(DATASET) ACCESS(READ) ID(*)
-
PERMIT 'FEK.#CUST.CNTL' CLASS(DATASET) ACCESS(READ) ID(*)
```

Note: No permits are needed for `FEK.SFEKLPA`, as all code that resides in LPA is accessible by everyone.

- permit JES Job Monitor to access the load and parameter library

```
-
PERMIT 'FEK.SFEKAUTH' CLASS(DATASET) ACCESS(READ) ID(STCJMON)
-
PERMIT 'FEK.#CUST.PARMLIB' CLASS(DATASET) ACCESS(READ) ID(STCJMON)
```

- (optional) permit clients to update the long/short name translation VSAM for SCLMDT
  –
  ```
  PERMIT 'FEK.#CUST.LSTRANS.*.**' CLASS(DATASET) ACCESS(UPDATE) ID(*)
  ```
- (optional) permit RAM developers to update the CARMA VSAMs for CARMA
  –
  ```
  PERMIT 'FEK.#CUST.CRA*.**' CLASS(DATASET) ACCESS(UPDATE) ID(#ram-developer)
  ```
- (optional) permit CICS users to read the CRD repository VSAM for Application Deployment Manager
  –
  ```
  PERMIT 'FEK.#CUST.ADNREP*.**' CLASS(DATASET) ACCESS(READ) ID(*)
  ```
- (optional) permit CICS administrators to update the CRD repository VSAM for Application Deployment Manager
  –
  ```
  PERMIT 'FEK.#CUST.ADNREP*.**' CLASS(DATASET) ACCESS(UPDATE) ID(#cicsadmin)
  ```
- (optional) permit CICS users to update the manifest repository VSAM for Application Deployment Manager
  –
  ```
  PERMIT 'FEK.#CUST.ADNMAN*.**' CLASS(DATASET) ACCESS(UPDATE) ID(*)
  ```
- (optional) permit CICS TS server to access the load library for bidi and Application Deployment Manager
  –
  ```
  PERMIT 'FEK.SFEKLOAD' CLASS(DATASET) ACCESS(READ) ID(#cicsts)
  ```
- (optional) permit DB2® server to access the exec library for DB2 stored procedure builder
  –
  ```
  PERMIT 'FEK.SFEKPROC' CLASS(DATASET) ACCESS(READ) ID(#db2)
  ```
- activate security profiles
  –
  ```
  SETROPTS GENERIC(DATASET) REFRESH
  ```

When controlling `READ` access to system data sets, you must provide Developer for System z servers and users permission to `READ` the following data sets:

- `CEE.SCEERUN`
- `CEE.SCEERUN2`
- `CBC.SCLBDLL`
- `ISP.SISPLOAD`
- `ISP.SISPLPA`
- `SYS1.LINKLIB`
- `SYS1.SIEALNKE`
- `REXX.V1R4M0.SEAGLPA`

Note: When you use the Alternate Library for REXX product package, the default REXX runtime library name is `REXX.*.SEAGALT`. instead of `REXX.*.SEAGLPA`, as used in the preceding list.

Define the Developer for System z started tasks

The following sample RACF commands create the JMON, RSED, and LOCKD started tasks, with protected user IDs (STCJMON, STCRSE, and STCLOCK respectively) and group STCGROUP assigned to them. Replace the #group-id and #user-id-* placeholders with valid OMVS IDs.

•
```
ADDGROUP STCGROUP OMVS(GID(#group-id))
DATA('GROUP WITH OMVS SEGMENT FOR STARTED TASKS')
```
•
```
ADDUSER STCJMON DFLTGROUP(STCGROUP) NOPASSWORD NAME('RDZ - JES JOBMONITOR')
OMVS(UID(#user-id-jmon) HOME(/tmp) PROGRAM(/bin/sh) NOASSIZEMAX
NOTHREADSMAX)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
```
•
```
ADDUSER STCRSE DFLTGROUP(STCGROUP) NOPASSWORD NAME('RDZ - RSE DAEMON')
OMVS(UID(#user-id-rse) HOME(/tmp) PROGRAM(/bin/sh) ASSIZEMAX(2147483647)
NOTHREADSMAX)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
```
•
```
ADDUSER STCLOCK DFLTGROUP(STCGROUP) NOPASSWORD NAME('RDZ - LOCK DAEMON')
OMVS(UID(#user-id-lock) HOME(/tmp) PROGRAM(/bin/sh) NOASSIZEMAX)
NOTHREADSMAX)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
```
•
```
RDEFINE STARTED JMON.* DATA('RDZ - JES JOBMONITOR')
STDATA(USER(STCJMON) GROUP(STCGROUP) TRUSTED(NO))
```
•
```
RDEFINE STARTED RSED.* DATA('RDZ - RSE DAEMON')
STDATA(USER(STCRSE) GROUP(STCGROUP) TRUSTED(NO))
```
•
```
RDEFINE STARTED LOCKD.* DATA('RDZ - LOCK DAEMON')
STDATA(USER(STCLOCK) GROUP(STCGROUP) TRUSTED(NO))
```
•
```
SETROPTS RACLIST(STARTED) REFRESH
```

Note:

- Ensure that the started tasks user IDs are protected by specifying the NOPASSWORD keyword.
- Ensure that RSE server has a unique OMVS uid due to the z/OS UNIX related privileges granted to this uid.
- RSE daemon requires a large address space size (2GB) for proper operation. You should set this value in the ASSIZEMAX variable of the OMVS segment for user ID STCRSE. This to ensure that RSE daemon will get the required region size, regardless of changes to MAXASSIZE in SYS1.PARMLIB(BPXPRMxx).
- RSE also requires a large number of threads for proper operation. You can set the limit in the THREADSMAX variable of the OMVS segment for user ID STCRSE. This ensures that RSE will get the required thread limit, regardless of changes to MAXTHREADS or MAXTHREADTASKS in SYS1.PARMLIB(BPXPRMxx). Refer to "Tuning considerations" in the *Host Configuration Reference* (SC14-7290) to determine the correct value for the thread limit.
- User ID STCJMON is another good candidate for setting THREADSMAX in the OMVS segment, because JES Job Monitor uses a thread per client connection.

You might want to consider making the STCRSE user ID restricted. Users with the RESTRICTED attribute cannot access protected (MVS) resources they are not specifically authorized to access.

```
ALTUSER STCRSE RESTRICTED
```

To ensure that restricted users do not gain access to z/OS UNIX file system resources through the "other" permission bits, you must define the RESTRICTED.FILESYS.ACCESS profile in the UNIXPRIV class with UACC(NONE). Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information about restricting user IDs.

Attention: If you use restricted user IDs, you must explicitly add the permission to access a resource with the TSO **PERMIT** or the z/OS UNIX **setfacl** commands. This includes resources where the Developer for System z documentation uses UACC (such as the ** profile in the PROGRAM class) or where it relies on common z/OS UNIX conventions (such as everyone having read and execute permission for Java libraries). Test this before activating it on a production system.

Define JES command security

JES Job Monitor issues all JES operator commands requested by a user through an extended MCS (EMCS) console, whose name is controlled with the CONSOLE_NAME directive, as documented in "FEJJCNFG, JES Job Monitor configuration file" in the *IBM Rational Developer for System z Host Configuration Guide*.

The following sample RACF commands give Developer for System z users conditional access to a limited set of JES commands (Hold, Release, Cancel, and Purge). Users only have execution permission if they issue the commands through JES Job monitor. Replace the #console placeholder with the actual console name.

- ```
  RDEFINE OPERCMDS MVS.MCSOPER.#console UACC(READ)
  DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
  ```
- ```
  RDEFINE OPERCMDS JES%.** UACC(NONE)
  ```
- ```
  PERMIT JES%.** CLASS(OPERCMDS) ACCESS(UPDATE) WHEN(CONSOLE(JMON)) ID(*)
  ```
- ```
  SETROPTS RACLIST(OPERCMDS) REFRESH
  ```

Note:

- Usage of the console is permitted if no MVS.MCSOPER.#console profile is defined
- The CONSOLE class must be active for WHEN(CONSOLE(JMON)) to work, but there is no actual profile check in the CONSOLE class for EMCS consoles.
- Do not replace JMON with the actual console name in the WHEN(CONSOLE(JMON)) clause. The JMON keyword represents the point-of-entry application, not the console name.

Attention: Defining JES commands with universal access NONE in your security software might impact other applications and operations. Test this before activating it on a production system.

Table 8 and Table 9 show the operator commands issued for JES2 and JES3, and the discrete security profiles that can be used to protect them.

Table 8. JES2 Job Monitor operator commands

Action	Command	OPERCMDS profile	Required access

Hold	`$Hx(jobid)`
	
	with x = {J, S or T}
		`jesname.MODIFYHOLD.BAT`
		`jesname.MODIFYHOLD.STC`
		`jesname.MODIFYHOLD.TSU`
			`UPDATE`

Release	`$Ax(jobid)`
	
	with x = {J, S or T}
		`jesname.MODIFYRELEASE.BAT`
		`jesname.MODIFYRELEASE.STC`
		`jesname.MODIFYRELEASE.TSU`
			`UPDATE`

Cancel	`$Cx(jobid)`
	
	with x = {J, S or T}
		`jesname.CANCEL.BAT`
		`jesname.CANCEL.STC`
		`jesname.CANCEL.TSU`
			`UPDATE`

Purge	`$Cx(jobid),P`
	
	with x = {J, S or T}
		`jesname.CANCEL.BAT`
		`jesname.CANCEL.STC`
		`jesname.CANCEL.TSU`
			`UPDATE`

Table 9. JES3 Job Monitor operator commands

Action	Command	OPERCMDS profile	Required access
Hold	`*F,J=jobid,H`	`jesname.MODIFY.JOB`	`UPDATE`
Release	`*F,J=jobid,R`	`jesname.MODIFY.JOB`	`UPDATE`
Cancel	`*F,J=jobid,C`	`jesname.MODIFY.JOB`	`UPDATE`
Purge	`*F,J=jobid,C`	`jesname.MODIFY.JOB`	`UPDATE`

Note:

- The Hold, Release, Cancel, and Purge JES operator commands, and the Show JCL command, can only be executed against spool files owned by the client user ID, unless `LIMIT_COMMANDS=` with value `LIMITED` or `NOLIMIT` is specified in the JES Job Monitor configuration file. Refer to "Actions against jobs - target limitations" in the *Host Configuration Reference* (SC14-7290) for more information about this.
- Users can browse any spool file, unless `LIMIT_VIEW=USERID` is defined in the JES Job Monitor configuration file. Refer to "Access to spool files" in *Host Configuration Reference* (SC14-7290) for more information about this.
- Without being authorized for these operator commands, users will still be able to submit jobs and read job output through JES Job Monitor, if they have sufficient authority to possible profiles that protect these resources (such as those in the `JESINPUT`, `JESJOBS` and `JESSPOOL` classes).

Assuming the identity of the JES Job Monitor server by creating a JMON console from a TSO session is prevented by your security software. Even though the console can be created, the point of entry is different (JES Job Monitor versus TSO). JES commands issued from this console will fail the security check, if your security is set up as documented in this publication and the user does not have authority to the JES commands through other means.

Define RSE as a secure z/OS UNIX server

RSE requires UPDATE access to the `BPX.SERVER` profile to create/delete the security environment for the client's thread. If this profile is not defined, `UID(0)` is required for RSE.

- RDEFINE FACILITY BPX.SERVER UACC(NONE)
- PERMIT BPX.SERVER CLASS(FACILITY) ACCESS(UPDATE) ID(STCRSE)
- SETROPTS RACLIST(FACILITY) REFRESH

Attention: Defining the BPX.SERVER profile makes z/OS UNIX as a whole switch from UNIX level security to z/OS UNIX level security, which is more secure. This might impact other z/OS UNIX applications and operations. Test this before activating it on a production system. Refer to *UNIX System Services Planning* (GA22-7800) for more information about the different security levels.

Define MVS program controlled libraries for RSE

Servers with authority to BPX.SERVER must run in a clean, program-controlled environment. This implies that all programs called by RSE must also be program controlled. For MVS load libraries, program control is managed by your security software.

RSE uses system (SYS1.LINKLIB), Language Environment®'s runtime (CEE.SCEERUN*) and ISPF's TSO/ISPF Client Gateway (ISP.SISPLOAD) load library.

- RALTER PROGRAM ** UACC(READ) ADDMEM('SYS1.LINKLIB'//NOPADCHK)
- RALTER PROGRAM ** UACC(READ) ADDMEM('CEE.SCEERUN'//NOPADCHK)
- RALTER PROGRAM ** UACC(READ) ADDMEM('CEE.SCEERUN2'//NOPADCHK)
- RALTER PROGRAM ** UACC(READ) ADDMEM('ISP.SISPLOAD'//NOPADCHK)
- SETROPTS WHEN(PROGRAM) REFRESH

Note: Do not use the ** profile if you already have a * profile in the PROGRAM class. It obscures and complicates the search path used by your security software. In this case, you must merge the existing * and the new ** definitions. IBM recommends using the ** profile, as documented in *Security Server RACF Security Administrator's Guide* (SA22-7683).

The following additional (prerequisite) libraries must be made program controlled to support the use of optional services. This list does not include data sets that are specific to a product that Developer for System z interacts with, such as IBM Debug Tool.

- Alternate REXX runtime library (for SCLM Developer Toolkit)
  - REXX.*.SEAGALT
- System load library (for SSL encryption)
  - SYS1.SIEALNKE
- File Manager listener load library (for File Manager integration)
  - FMN.SFMNMODA

Note: Libraries that are designed for LPA placement also require program control authorizations if they are accessed through LINKLIST or STEPLIB. This publication documents the usage of the following LPA libraries:

- ISPF (for TSO/ISPF Client Gateway)
  - ISP.SISPLPA
- REXX runtime library (for SCLM Developer Toolkit)
  - REXX.*.SEAGLPA
- Developer for System z (for CARMA)

– FEK.SFEKLPA

Define application protection for RSE

During client logon, RSE daemon verifies that a user is allowed to use the application.

•
```
RDEFINE APPL FEKAPPL UACC(READ) DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
```
•
```
SETROPTS RACLIST(APPL) REFRESH
```

Note:

- As described in more detail in "Define PassTicket support for RSE," RSE supports the usage of an application ID other than FEKAPPL. The APPL class definition must match the actual application ID used by RSE.
- The client connection request will succeed if the application ID is not defined in the APPL class.
- The client connection request will only fail if the application ID is defined and the user lacks READ access to the profile.

Define PassTicket support for RSE

The client's password (or other means of identification, such as an X.509 certificate) is only used to verify his identity upon connection. Afterwards, PassTickets are used to maintain thread security.

PassTickets are system-generated passwords with a lifespan of about 10 minutes. The generated PassTickets are based upon a secret key. This key is a 64 bit number (16 hex characters). In the following sample RACF commands, replace the key16 placeholder with a user-supplied 16 character hex string (characters 0-9 and A-F).

•
```
RDEFINE PTKTDATA FEKAPPL UACC(NONE) SSIGNON(KEYMASKED(key16))
APPLDATA('NO REPLAY PROTECTION – DO NOT CHANGE')
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
```
•
```
RDEFINE PTKTDATA IRRPTAUTH.FEKAPPL.* UACC(NONE)
DATA('RATIONAL DEVELOPER FOR SYSTEM Z')
```
•
```
PERMIT IRRPTAUTH.FEKAPPL.* CLASS(PTKTDATA) ACCESS(UPDATE) ID(STCRSE)
```
•
```
SETROPTS RACLIST(PTKTDATA) REFRESH
```

RSE supports the usage of an application ID other than FEKAPPL. Uncomment and customize the "APPLID=FEKAPPL" option in rsed.envvars to activate this, as documented in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *IBM Rational Developer for System z Host Configuration Guide*. The PTKTDATA class definitions must match the actual application ID used by RSE.

You should not use OMVSAPPL as application ID, because it will open the secret key to most z/OS UNIX applications. You should also not use the default MVS application ID, which is MVS followed by the system's SMF ID, because this will open the secret key to most MVS applications (including user batch jobs).

Note:

- If the PTKTDATA class is already defined, verify that it is defined as a generic class before creating the profiles listed previously. The support for generic characters in the PTKTDATA class is new since z/OS release 1.7, with the introduction of a Java interface to PassTickets.
- Substitute the wildcard (*) in the IRRPTAUTH.FEKAPPL.* definition with a valid user ID mask to limit the user IDs for which RSE can generate a PassTicket.
- Depending on your RACF settings, the user defining a profile may also be on the access list of the profile. It is advised that you remove this permission for the PTKTDATA profiles.
- JES Job Monitor and RSE must have the same application ID to allow JES Job Monitor to evaluate the PassTickets presented by RSE.
- If the system has a cryptographic product installed and available, you can encrypt the secured signon application key for added protection. In order to do so, use the KEYENCRYPTED keyword instead of KEYMASKED. Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information about this.

Attention: The client connection request will fail if PassTickets are not set up correctly.

Define z/OS UNIX program controlled files for RSE

Servers with authority to BPX.SERVER must run in a clean, program-controlled environment. This implies that all programs called by RSE must also be program controlled. For z/OS UNIX files, program control is managed by the **extattr** command. To execute this command, you need READ access to BPX.FILEATTR.PROGCTL in the FACILITY class, or be UID(0).

RSE server uses RACF's Java shared library (/usr/lib/libIRRRacf*.so).

- `extattr +p /usr/lib/libIRRRacf*.so`

Note:

- Since z/OS 1.9, /usr/lib/libIRRRacf*.so is installed program controlled during SMP/E RACF install.
- Since z/OS 1.10, /usr/lib/libIRRRacf*.so is part of SAF, which ships with base z/OS, so it is available also to non-RACF customers.
- The setup might be different if you use a security product other than RACF. Consult the documentation of your security product for more information.
- The SMP/E install of Developer for System z sets the program control bit for internal RSE programs.
- Use the **ls -Eog** z/OS UNIX command to display the current status of the program control bit (the file is program controlled if the letter **p** shows in the second string).

```
$ ls -Eog /usr/lib/libIRRRacf*.so
-rwxr-xr-x  aps-  2     69632 Oct  5  2007 /usr/lib/libIRRRacf.so
-rwxr-xr-x  aps-  2     69632 Oct  5  2007 /usr/lib/libIRRRacf64.so
```

Verify security settings

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

- Security settings and classes
  - SETROPTS LIST
- OMVS segment for users

  - `LISTUSER #userid NORACF OMVS`
  - `LISTGRP #group-name NORACF OMVS`
- Data set profiles
  - `LISTGRP FEK`
  - `LISTDSD PREFIX(FEK) ALL`
- Started tasks
  - `LISTGRP  STCGROUP OMVS`
  - `LISTUSER STCJMON  OMVS`
  - `LISTUSER STCRSE   OMVS`
  - `LISTUSER STCLOCK  OMVS`
  - `RLIST STARTED JMON.*  ALL STDATA`
  - `RLIST STARTED RSED.*  ALL STDATA`
  - `RLIST STARTED LOCKD.* ALL STDATA`
- JES command security
  - `RLIST CONSOLE JMON ALL`
  - `RLIST OPERCMDS MVS.MCSOPER.JMON ALL`
  - `RLIST OPERCMDS JES%.** ALL`
- RSE as a secure z/OS UNIX server
  - `RLIST FACILITY BPX.SERVER ALL`
- MVS program controlled libraries for RSE
  - `RLIST PROGRAM ** ALL`
- Application protection for RSE
  - `RLIST APPL FEKAPPL ALL`
- PassTicket support for RSE
  - `RLIST PTKTDATA FEKAPPL ALL SSIGNON`
  - `RLIST PTKTDATA IRRPTAUTH.FEKAPPL.* ALL`
- z/OS UNIX program controlled files for RSE
  - `ls -E /usr/lib/libIRRacf*.so`

Chapter 3. TCP/IP considerations

Developer for System z 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

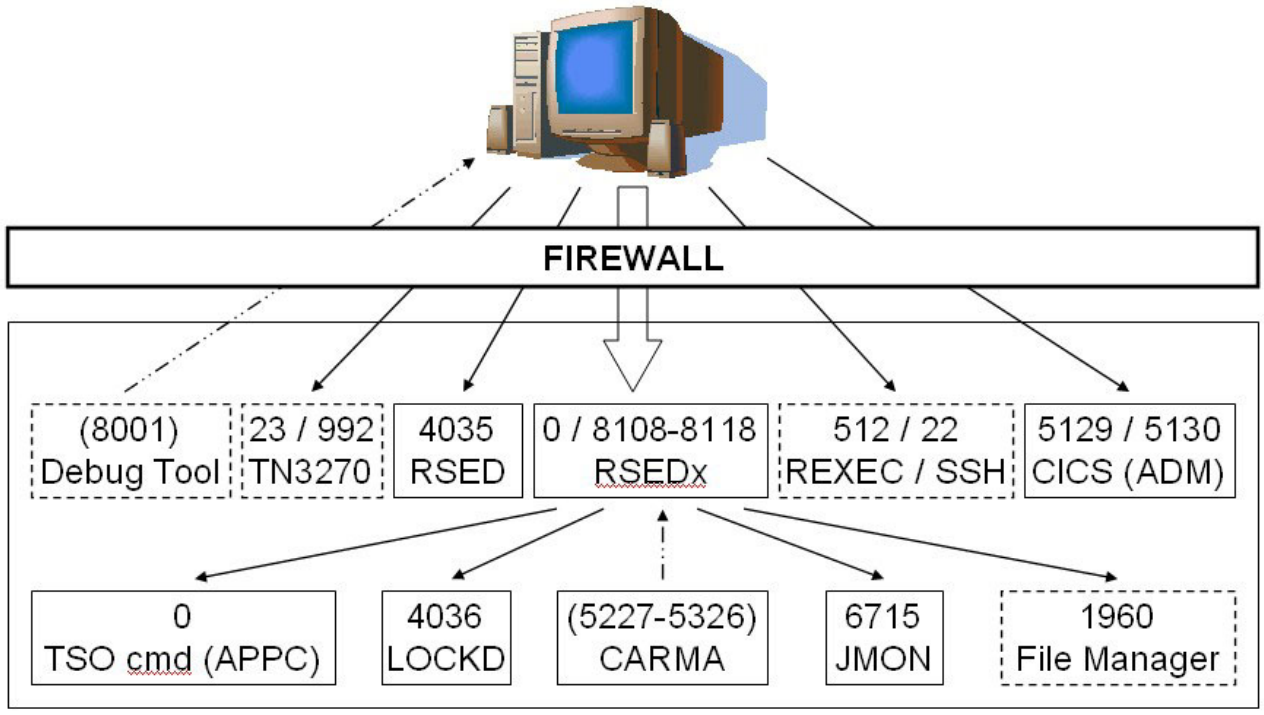


Figure 8. TCP/IP ports

Figure 8 shows the TCP/IP ports that can be used by Developer for System z. 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):

- RSE daemon for client-host communication setup, default port `4035`. Communication on this port can be encrypted using SSL.
- 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

rsed.envvars. The default port range for _RSE_PORTRANGE is 8108-8118 (11 ports). Communication on this port can be encrypted using SSL.

- (optional) 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 using SSL.
- (optional) TN3270 Telnet service for the Host Connect Emulator, default port 23. Communication can be encrypted using SSL (default port 992). The default port assigned to the TN3270 Telnet service depends on whether or not the user chooses to use encryption.
- (optional) Either or both CICSTS application interfaces for Application Deployment Manager:
  - RESTful interface, default port 5130.
  - Web Services interface, default port 5129. Communication on this port can be encrypted using SSL.

Note: During a remote debug session for Cobol, PL/I or Assembler, IBM Debug Tool for z/OS is invoked. This product communicates directly with the client. This communication is initiated on the host, and connects to port 8001 on the client.

Internal communication

Several Developer for System z host services run in separate threads or address spaces and are using TCP/IP sockets as communication mechanism. 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:

- JES Job Monitor for JES-related services, default port 6715. The port can be set in the FEJJCNFG configuration member and is repeated in the rsed.envvars configuration member.
- Lock daemon for data set lock-related services, default port 4036. The port can be set in the rsed.envvars configuration member.
- (optional) File Manager Integration for interacting with IBM File Manager, default port 1960. The port is set during File Manger customization and is repeated in the FMIEXT.properties configuration member.
- (optional) CARMA communication, default port range 5227-5326 (100 ports). The port range can be set in the CRASRV.properties configuration file.
- (optional) The APPC version of the TSO Commands service uses any socket available to communicate with the lock manager (which enqueues MVS data sets for clients). You cannot set a specific port range to be used.

TCP/IP port reservation

If you use the PORT or PORTRANGE statement in PROFILE.TCPIP to reserve the ports used by Developer for System z, note that many binds are done by threads active in a 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  ; Developer for System z - RSE daemon
PORT      4036     TCP LOCKD ; Developer for System z - lock daemon
PORT      6715     TCP JMON  ; Developer for System z - JES job monitor
PORTRange 8108 11  TCP RSED* ; Developer for System z - _RSE_PORTRANGE
PORTRange 5227 100 TCP RSED* ; Developer for System z - CARMA
```

CARMA and TCP/IP ports

CARMA (Common Access Repository Manager) is used to access a host based Software Configuration Manager (SCM), for example CA Endevor® 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 a RSE thread pool, will 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. Once the server is connected, the client can send requests to the server and receive the results.

So 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 a 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 System z - CARMA
```

LDAP considerations

RSE server can be configured to query one or more LDAP servers for various Developer for System z services:

- Query LDAP groups for push-to-client multiple developer group support.
- Query one or more Certificate Revocation Lists (CRLs) for X.509 authentication.

Note that TCP/IP security measures, such as firewalls, might stop the (host-based) RSE server from contacting the LDAP server. Use the following information to ensure the LDAP server can be reached:

- The LDAP server TCP/IP addresses or DNS names are listed in `*_LDAP_SERVER` variables in `rsed.envvars`.
- The LDAP server port numbers are listed in `*_LDAP_PORT` variables in `rsed.envvars`.
- LDAP uses the TCP protocol.
- The LDAP server is contacted by the host-based RSE server.
- RSE server is active in an `RSEDx` address space, where `RSED` is the name of the RSE started task and `x` is a random one-digit number, for example `RSED8`.

Overriding default TCP/IP behavior

Delayed ACK

Delayed ACK delays the receipt acknowledgement (ACK) of a TCP packet by up to 200ms. This delay increases the chance that the ACK can be sent along with the response to the received packet, reducing network traffic. However, if the sender is waiting for the ACK before sending a new packet (for example, due to implementation of Nagle's algorithm), and there is no response to the packet just sent (for example, because it is part of a file transfer), communication is unnecessary delayed.

Developer for System z allows you to disable the delayed ACK function. On the host, this is done with the DSTORE_TCP_NO_DELAY directive in rsed.envvars, as documented in the *Host Configuration Guide* (SC23-7658).

Multi-stack (CINET)

z/OS Communication Server allows you to have multiple TCP/IP stacks concurrently active on a single system. This is referred to as a CINET setup.

If Developer for System z is not active on the default stack, then selected Developer for System z functions might fail. Using stack affinity is a sure way to resolve this. Stack affinity instructs Developer for System z to use only a specific TCP/IP stack (instead of every available TCP/IP stack, which is the default for the started tasks).

Stack affinity is set for the Developer for System z started tasks by uncommenting and customizing the _BPXK_SETIBMOPT_TRANSPORT directive in the FEJJCNFG and rsed.envvars configuration files. Refer to the related sections in "Chapter 2 Basic Customization" of the *Host Configuration Guide* (SC23-7658) for more details on customizing these configuration files.

CARMA and stack affinity

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

Similar to the Developer for System z 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* (SC23-7658) 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.

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.

Distributed Dynamic VIPA

Distributed DVIPA (Dynamic Virtual IP Addressing) allows you to concurrently run identical Developer for System z setups on different systems in your sysplex, and have TCP/IP, optionally with the help of WLM, distribute the client connections among these systems.

There are several ways you can configure a distributed DVIPA, but Developer for System z does impose some restrictions on these options.

- RSE daemon owns the port that is defined for distributed DVIPA, but the actual work happens in the RSE server, which is active as a thread in another address space. Therefore, you cannot use the SERVERWLM distribution method to do load balancing across your systems, because WLM will give advice based on statistics for RSE daemon, not RSE server.
- The client only knows the DVIPA address used by the Sysplex Distributor for RSE daemon. The Sysplex Distributor will pass the connection request to one of the available RSE daemons, which in turn will start an RSE server thread that will bind to a port on that system. When the client connects to this port, it will use the DVIPA address again, not the actual system address, so you must ensure that the Sysplex Distributor redirects the new connection to the correct system.

  Therefore, Developer for System z requires the definition of SYSPLEXPORTS in the VIPADISTRIBUTE statement to ensure that the ports used by the RSE server threads are unique within the sysplex.

  Note:
  - The usage of SYSPLEXPORTS implies that the EZBEPORT structure must be defined in your coupling facility.
  - The usage of SYSPLEXPORTS implies that TCP/IP will select an ephemeral port. This implies that you cannot reserve ports for these connections in your TCP/IP profile with the PORT and PORTRANGE directives.

There are also some restrictions within Developer for System z when using distributed DVIPA:

- To ensure that the Developer for System z client will not interfere with the correct port selection by TCP/IP, you should enable the deny.nonzero.port directive in rsed.envvars.
- The usage of SYSPLEXPORTS implies that TCP/IP will ensure that a unique port is used for each connection. This implies that when using _RSE_PORTRANGE in rsed.envvars, you must specify a range that is big enough to hold all your concurrently active users. In a single system setup, Developer for System z tends to use the same port for all connections, so the default range is small.

- All participating Developer for System z servers must have an identical setup. You should share /usr/lpp/rdz and /etc/rdz among all participating systems. You should also share /var/rdz/projects, /var/rdz/pushtoclient, and /var/rdz/sclmdt, if these directories are used. Note that /var/rdz/WORKAREA and /var/rdz/logs must be unique for each system.

JES Job Monitor, Lock daemon, CARMA and other Developer for System z servers only interact with the local RSE, and thus do not require a DVIPA setup.

Distributed DVIPAs are defined by the VIPADEFine and VIPABackup keywords of the VIPADynamic block in your TCP/IP profile. The VIPADISTribute keyword adds the required Sysplex Distributor definitions. Distributed DVIPA requires that all participating stacks are sysplex-aware, which is done via the SYSPLEXRouting and DYNAMICXCF keywords of the IPCONFIG block in your TCP/IP profile. Refer to *Communications Server: IP Configuration Reference* (SC31-8776) for more details on these directives.

Refer to *MVS Setting Up a Sysplex* (SA22-7625) and *Communication Server: SNA Network Implementation Guide* (SC31-8777) for more information on setting up the EZBEPORTS structure in your coupling facility.

Sample setup

In the following sample setup there are two z/OS systems, SYS1 and SYS2, which are part of a sysplex. System SYS1 is defined as the system that normally hosts the Sysplex Distributor for the Developer for System z distributed DVIPA.

After defining the distributed DVIPA, Developer for System z can be started on the systems to allow load balancing client connections across the systems. JES Job Monitor and the Lock daemon only interact with the local RSE, and thus do not require a DVIPA setup. Clients will connect to port 4035 on IP address 10.10.10.1.

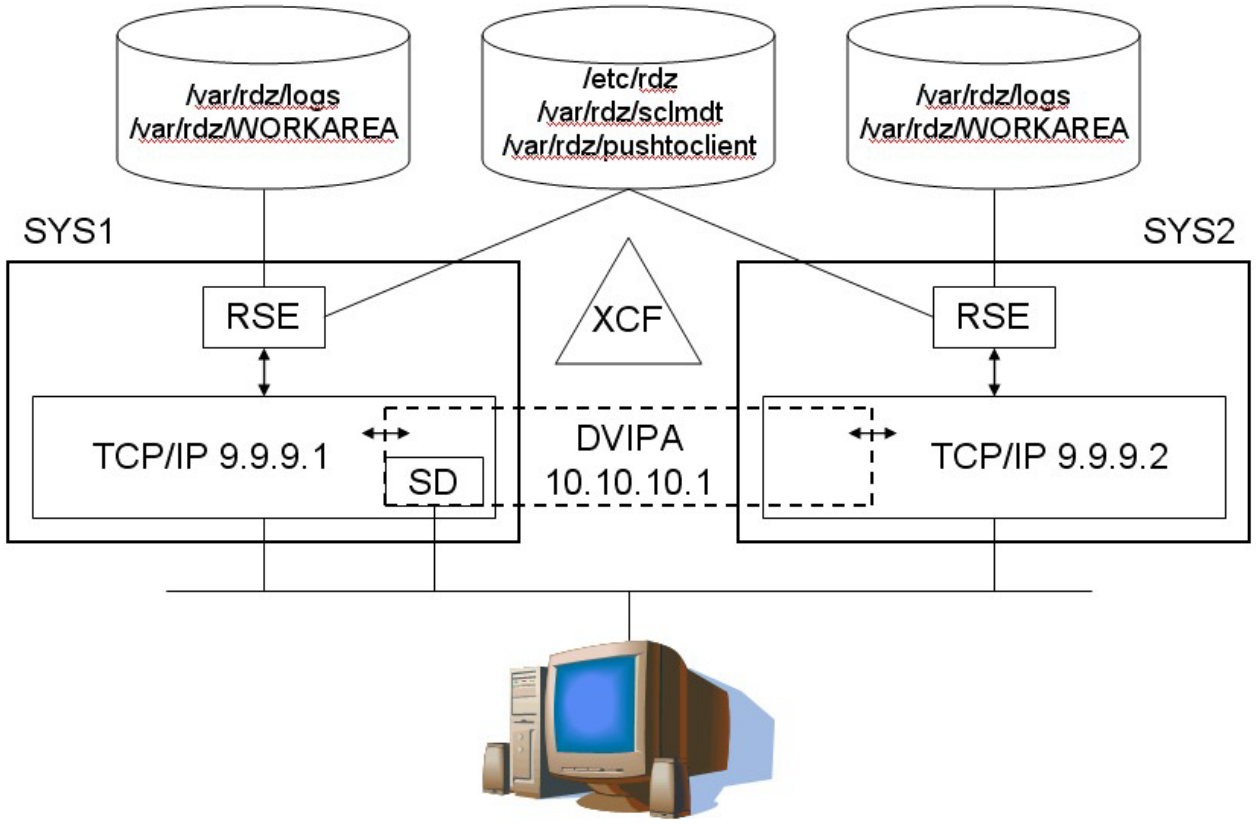


Figure 9. Distributed Dynamic VIPA sample

System SYS1 – TCP/IP profile

```
IPCONFIG
  SYSPLEXRouting
; SYSPLEXROUTING is required as this stack needs sysplex communication
  DYNAMICXCF 9.9.9.1   255.255.255.0   1
; DYNAMICXCF defines device/link with home address 9.9.9.1 as needed
  IGNORERedirect

VIPADYNAMIC
  VIPADEFINE 255.255.255.0   10.10.10.1
; VIPADEFINE defines 10.10.10.1 as main DVIPA on SYS1 for RDz
  VIPADISTRIBUTE DEFINE
; VIPADISTRIBUTE makes 10.10.10.1 a distributed DVIPA, must match SYS2
    SYSPLEXPORTS                  ; RDz prereq
    DISTMETHOD ROUNDROBIN         ;
    10.10.10.1                    ; DVIPA address used by RDz clients
    PORT 4035                     ; port used by RDz clients
    DESTIP 9.9.9.1   9.9.9.2      ; RDz active on SYS1 and SYS2
ENDVIPADYNAMIC
```

System SYS2 – TCP/IP profile

```
IPCONFIG
  SYSPLEXRouting
; SYSPLEXROUTING is required as this stack needs sysplex communication
  DYNAMICXCF 9.9.9.2   255.255.255.0   1
; DYNAMICXCF defines device/link with home address 9.9.9.2 as needed
  IGNORERedirect

VIPADYNAMIC
```

```
  VIPABACKUP 255.255.255.0   10.10.10.1
; VIPABACKUP defines 10.10.10.1 as backup DVIPA on SYS2 for RDz
  VIPADISTRIBUTE DEFINE
; VIPADISTRIBUTE makes 10.10.10.1 a distributed DVIPA, must match SYS1
    SYSPLEXPORTS                   ; RDz prereq
    DISTMETHOD ROUNDROBIN          ;
    10.10.10.1                     ; DVIPA address used by RDz clients
    PORT 4035                      ; port used by RDz clients
    DESTIP 9.9.9.1   9.9.9.2       ; RDz active on SYS1 and SYS2
ENDVIPADYNAMIC
```

Chapter 4. WLM considerations

Unlike traditional z/OS applications, Developer for System z is not a monolithic application that can be identified easily to Workload Manager (WLM). Developer for System z 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 System z," on page 1, 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

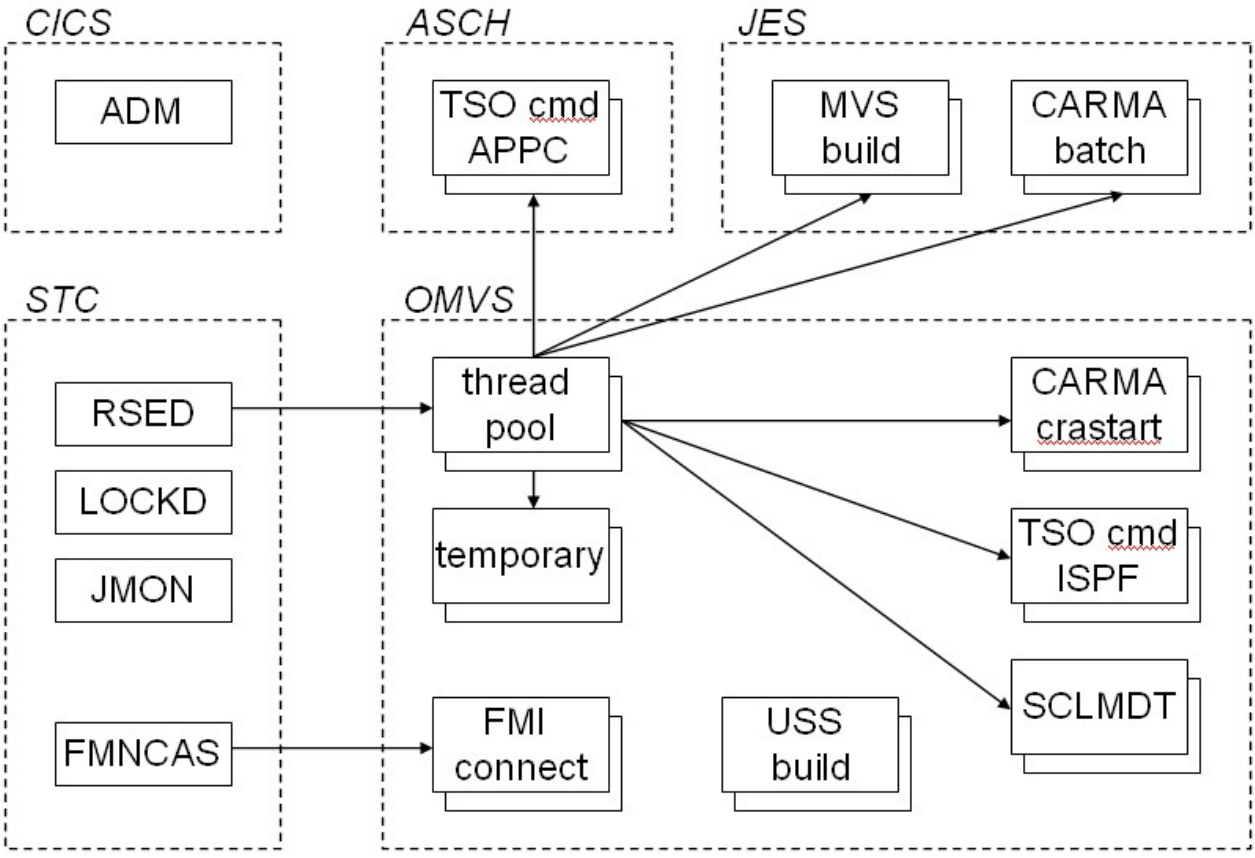


Figure 10. WLM classification

Figure 10 shows a basic overview of the subsystems through which Developer for System z workloads are presented to WLM.

Application Deployment Manager (ADM) is active within a CICS region, and will therefore follow the CICS classification rules in WLM.

RSE daemon (RSED), Lock daemon (LOCKD) and JES Job Monitor (JMON) are Developer for System z started tasks (or long-running batch jobs), each with their individual address space.

As documented in "RSE as a Java application" on page 3, 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 System z, some workloads, such as the TSO Commands service (TSO cmd) or CARMA, can run in different subsystems.

The address spaces listed in Figure 10 on page 55 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 System z uses, such as File Manager (FMNCAS) or 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 10 lists the subsystem types that can receive Developer for System z workloads.

Table 10. WLM entry-point subsystems

Subsystem type	Work description
ASCH	The work requests include all APPC transaction programs scheduled by the IBM-supplied APPC/MVS transaction scheduler, ASCH.
CICS	The work requests include all transactions processed by CICS.
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 11 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.

Table 11. WLM work qualifiers

		ASCH	CICS	JES	OMVS	STC
AI	Accounting Information	x		x	x	x

Table 11. WLM work qualifiers (continued)

		ASCH	CICS	JES	OMVS	STC
LU	LU Name (*)		x			
PF	Perform (*)			x		x
PRI	Priority			x		
SE	Scheduling Environment Name			x		
SSC	Subsystem Collection Name			x		
SI	Subsystem Instance (*)		x	x		
SPM	Subsystem Parameter					x
PX	Sysplex Name	x	x	x	x	x
SY	System Name (*)	x			x	x
TC	Transaction/Job Class (*)	x		x		
TN	Transaction/Job Name (*)	x	x	x	x	x
UI	User ID (*)	x	x	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 55, Developer for System z 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 System z 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 System z itself, not the whole system.

Table 12 lists the address spaces that are used by Developer for System z. z/OS UNIX will substitute "x" in the "Task Name" column by a random 1-digit number.

Table 12. WLM workloads

Description	Task name	Workload
JES Job Monitor	JMON	STC
Lock daemon	LOCKD	STC
RSE daemon	RSED	STC

Table 12. WLM workloads (continued)

Description	Task name	Workload
RSE thread pool	RSEDx	OMVS
ISPF Client Gateway (TSO Commands service and SCLMDT)	<userid>x	OMVS
TSO Commands service (APPC)	FEKFRSRV	ASCH
CARMA (batch)	CRA<port>	JES
CARMA (crastart)	<userid>x	OMVS
CARMA (ISPF Client Gateway)	<userid> and <userid>x	OMVS
MVS build (batch job)	*	JES
z/OS UNIX build (shell commands)	<userid>x	OMVS
z/OS UNIX shell	<userid>	OMVS
File Manager task	<userid>x	OMVS
Application Deployment Manager	CICSTS	CICS

Considerations for goal selection

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

- 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.

STC

All Developer for System z started tasks, RSE daemon, Lock daemon and JES Job Monitor, are servicing real-time client requests.

Table 13. WLM workloads - STC

Description	Task name	Workload
JES Job Monitor	JMON	STC
Lock daemon	LOCKD	STC
RSE daemon	RSED	STC

- JES Job Monitor

  JES Job Monitor provides all JES related services such as submitting jobs, browsing spool files and executing JES operator commands. 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 to moderate.
- Lock daemon

  The lock daemon queries the GRS enqueue tables upon client and operator request, and matches the result against known Developer for System z users. You should specify a high-performance, one-period velocity goal, because the task does not report individual transactions to WLM. Resource usage is expected to be minimal.
- RSE daemon

  RSE daemon handles client logon and authentication, and manages the different RSE thread pools. You should specify a high-performance, one-period velocity goal, because the task does not report individual transactions to WLM. Resource usage is expected to be moderate, with a peak at the beginning of the workday.

OMVS

The OMVS workloads can be divided into two groups, RSE thread pools and everything else. This because all workloads, except RSE thread pools, 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.)

Table 14. WLM workloads - OMVS

Description	Task name	Workload
RSE thread pool	RSEDx	OMVS
ISPF Client 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
File Manager task	<userid>x	OMVS

- RSE thread pool

An RSE thread pool is like the heart and brain of Developer for System z. Almost all data flows through here, and the miners (user specific threads) inside the thread pool control the actions of most other Developer for System z related tasks. 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 substantial.

The remaining 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.

- ISPF Client Gateway

  The ISPF Client Gateway is an ISPF service invoked by Developer for System z to execute non-interactive TSO and ISPF commands. This includes explicit commands issued by the client as well as implicit commands issued by Developer for System z, such as getting a PDS member list. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.
- CARMA

  CARMA is an optional Developer for System z server that is used to interact with host based Software Configuration Managers (SCMs), such as CA Endevor® SCM. Developer for System z 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.
- IBM File Manager

  Although not Developer for System z address spaces, the spawned File Manager child processes are listed here because they can be started upon request of a Developer for System z client, and these tasks use the same naming convention as Developer for System z tasks. These File Manager tasks process non-trivial MVS data set actions, such as formatted editing of a VSAM file. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal to moderate.

JES

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

Table 15. WLM workloads - JES

Description	Task name	Workload
CARMA (batch)	CRA<port>	JES
MVS build (batch job)	*	JES

- CARMA

  CARMA is an optional Developer for System z server that is used to interact with host based Software Configuration Managers (SCMs), such as CA Endevor® SCM. Developer for System z 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 System z 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.

ASCH

In the current Developer for System z versions, the ISPF Client Gateway is used to execute non-interactive TSO and ISPF commands. Due to historical reasons, Developer for System z also supports executing these commands via an APPC transaction. You should note that the APPC method is deprecated.

Table 16. WLM workloads - ASCH

Description	Task name	Workload
TSO Commands service (APPC)	FEKFRSRV	ASCH

- TSO Commands service

  The TSO Commands service can be started as an APPC transaction by Developer for System z to execute non-interactive TSO and ISPF commands. This includes explicit commands issued by the client as well as implicit commands issued by Developer for System z, such as getting a PDS member list. You should specify a multi-period goal for this service class. For the first periods, you should specify high-performance, percentile response time goals. For the last period, you should specify a moderate-performance velocity goal. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

CICS

Application Deployment Manager is an optional Developer for System z server that is active inside a CICS Transaction Server region.

Table 17. WLM workloads - CICS

Description	Task name	Workload
Application Deployment Manager	CICSTS	CICS

- Application Deployment Manager

  The optional Application Deployment Manager server, which is active inside a CICSTS region, allows you to securely offload selected CICSTS management tasks to developers. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal. The type of service class you should use depends on the other transactions active in this CICS region, and is therefore not discussed in detail.

WLM supports multiple types of management that you can use for CICS:

- Managing CICS toward a region goal

  The goal is set to a service class that manages the CICS address spaces. You can only use an execution velocity goal for this service class. WLM uses the JES or STC classification rules for the address spaces but does not use the CICS subsystem classification rules for transactions.
- Managing CICS toward a transaction response time goal

  A response time goal can be set in a service class assigned to a single transaction or a group of transactions. WLM uses the JES or STC classification rules for the address spaces and the CICS subsystem classification rules for transactions.

Chapter 5. Tuning considerations

As explained in Chapter 1, "Understanding Developer for System z," on page 1, RSE (Remote Systems Explorer) is the core of Developer for System z. To manage the connections and workloads from the clients, RSE is composed of a daemon address space, which controls thread pooling address spaces. The daemon acts as a focal point for connection and management purposes, while the thread pools process the client workloads.

This makes RSE a prime target for tuning the Developer for System z setup. However, maintaining hundreds of users, each using 16 or more threads, a certain amount of storage, and possibly 1 or more address spaces requires proper configuration of both Developer for System z and z/OS.

The following topics are covered in this chapter:

Resource usage

Use the information in this section to estimate the normal and maximum resource usage by Developer for System z, so you can plan your system configuration accordingly.

When you use the numbers and formulas presented in this section to define the values for system limits, be aware that you are working with fairly accurate estimates. Leave enough margin when setting the system limits to allow resource usage by temporary and other tasks, or by users connecting multiple times to the host simultaneously. (For example, by way of RSE and TN3270).

Note:

- The information is limited in scope to services accessed through RSE that are provided by Developer for System z itself. For example, resource usage of TN3270 is not documented (not accessed through RSE), nor is the resource usage of the programs called during remote (host-based) builds of MVS or z/OS UNIX projects (not provided by Developer for System z).
- Adding third-party extensions to Developer for System z can increase the resource usage counters.
- All services have short-lived "housekeeping" tasks, which use resources during their execution, and which may run sequential or parallel to each other. The resources used by these tasks are not documented.
- Where useful, user-specific resource usage of requisite software, such as the ISPF Client Gateway, is documented.
- The numbers presented here can change without prior notification.

Overview

The following tables give an overview of the number of address spaces, processes, and threads used by Developer for System z. More details on the numbers presented here can be found in the next sections:

- "Address space count" on page 65
- "Process count" on page 68
- "Thread count" on page 70

Table 18 gives a general overview of the key resources used by the Developer for System z started tasks. These resources are allocated only once. They are shared among all Developer for System z clients.

Table 18. Common resource usage

Started task	Address spaces	Processes	Threads
JMON	1	1	3
LOCKD	1	3	10
RSED	1	3	11
RSEDx	1 + (a)	1 + 2	10

Note: (a) There is one APF-authorized address space and at least 1 RSE thread pool address space active. Refer to "Address space count" on page 65 to determine the actual number of RSE thread pool address spaces.

Table 19 gives a general overview of the key resources used by requisite software. These resources are allocated for each Developer for System z client that invokes the related function.

Table 19. User-specific requisite resource usage

Requisite software	Address spaces	Processes	Threads
ISPF Client Gateway	1	2	4
APPC	1	1	2
File Manager	1	1	2

Table 20 gives a general overview of the key resources used by each Developer for System z client when executing the specified function. Non-numeric values, such as ISPF, are a reference to the corresponding value in Table 19.

Table 20. User-specific resource usage

User action	Address spaces	Processes	Threads		
	User ID	User ID	User ID	RSEDx	JMON
Logon	-	-	-	16	1
Timer for idle timeout	-	-	-	1	-
Expand PDS(E)	ISPF	ISPF	ISPF	-	-
Open data set	ISPF	ISPF	ISPF	-	-

Table 20. User-specific resource usage (continued)

User action	Address spaces	Processes	Threads		
	User ID	User ID	User ID	RSEDx	JMON
TSO command	ISPF	ISPF	ISPF	-	-
z/OS UNIX shell	1	1	1	6	-
MVS build	1	-	-	-	-
z/OS UNIX build	3	3	3	-	-
CARMA (batch)	1	1	2	1	-
CARMA (crastart)	1	1	2	4	-
CARMA (ispf)	4	4	7	5	-
SCLMDT	ISPF	ISPF	ISPF	-	-
File Manager Integration	ISPF + FM	ISPF + FM	ISPF + FM	-	-
Fault Analyzer Integration	-	-	-	-	-

Note: ISPF can be substituted by APPC, except for SCLM Developer Toolkit.

Address space count

Table 21 lists the address spaces that are used by Developer for System z, where "u" in the "Count" column indicates that the amount must be multiplied by the number of concurrently active users using the function. z/OS UNIX will substitute "x" in the "Task Name" column by a random 1-digit number.

Table 21. Address space count

Count	Description	Task name	Shared	Ends after
1	JES Job Monitor	JMON	Yes	Never
1	Lock daemon	LOCKD	Yes	Never
1	RSE daemon	RSED	Yes	Never
1	RSE APF authorized	RSEDx	Yes	Never
(a)	RSE thread pool	RSEDx	Yes	Never
lu	ISPF Client Gateway (TSO Commands service and SCLMDT)	<userid>x	No	15 minutes or user logoff
lu	TSO Commands service (APPC)	FEKFRSRV	No	60 minutes or user logoff
lu	CARMA (batch)	CRA<port>	No	7 minutes or user logoff
lu	CARMA (crastart)	<userid>x	No	7 minutes or user logoff
4u	CARMA (ispf)	(1)<userid> or (3)<userid>x	No	7 minutes or user logoff
(b)	Simultaneous ISPF Client Gateway usage by 1 user	<userid>x	No	Task completion

Table 21. Address space count (continued)

Count	Description	Task name	Shared	Ends after
1u	MVS build (batch job)	*	No	Task completion
3u	z/OS UNIX build (shell commands)	<userid>x	No	Task completion
1u	z/OS UNIX shell	<userid>	No	User logoff
(c)	File Manager	<userid>x	No	Task completion

Note:

- (a) There is at least one RSE thread pool address space active. The actual number depends on:
  - The `minimum.threadpool.process` directive in `rsed.envvars`. The default value is 1.
  - The number of users that can be serviced by one thread pool. The default settings aim for 60 users per thread pool.

  Note: If the `single.logon` directive is active, then there will be at least 2 thread pools started, even if `minimum.threadpool.process` is set to 1. The default setting for `single.logon` in `rsed.envvars` is active.
- (b) Developer for System z has multiple threads active per user. In the event that the ISPF Client Gateway address space has not finished serving the request of one thread when another thread sends a request, ISPF will start up a new Client Gateway to process the new request. This address space ends after task completion.
- (c) The File Manager listener starts an address space per object that must be manipulated, for example a VSAM. This address space stays active until Developer for System z signals that the object is no longer needed, for example by closing the VSAM.
- SCLMDT requires an ISPF Client Gateway address space. SCLMDT shares the address space with the TSO Commands service.
- Most MVS data set-related actions use the TSO Commands service, which can be active in the ISPF Client Gateway or an APPC transaction, respectively.

Use the formula in Figure 11 to estimate the maximum number of address spaces used by Developer for System z.

$$4 + A + N*(x + y + z) + (2 + N*0.01)$$

Figure 11. Maximum number of address spaces

Where

- "4" equals the number of permanent active server address spaces.
- "A" represents the number of RSE thread pool address spaces.
- "N" represents the maximum number of concurrent users.
- "x" is one of the following values, depending on the selected configuration options.

X	SCLMDT	TSO by way of Client Gateway	TSO by way of APPC
1	No	No	Yes
1	No	Yes	No
1	Yes	Yes	No

- "y" is one of the following values, depending on the selected configuration options.

Y	
0	No CARMA
1	CARMA (batch)
1	CARMA (crastart)
4	CARMA (ispf)

- "z" is 0 by default, but can increase depending on user actions:
  - Add 1 when a MVS build is performed. These address spaces end when the related build task (a batch job) completes.
  - Add 3 when a z/OS UNIX build is performed. Note that the actual number may be higher, depending on the needs of the programs invoked. These address spaces end when the related build task completes.
  - Add 1 for each concurrent interaction with IBM File Manager. These address spaces end when the requested object is no longer needed.
- "2 + N*0.01" adds a buffer for temporary address spaces. The required buffer size might differ at your site.

Use the formula in Figure 12 to estimate the maximum number of address spaces used by a Developer for System z client (not counting the undocumented temporary address spaces).

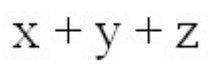


Figure 12. Number of address spaces per client

Where

- "x" depends on the selected configuration options and is documented for the formula to calculate the maximum number of address spaces (Figure 11 on page 66).
- "y" depends on the selected configuration options and is documented for the formula to calculate the maximum number of address spaces (Figure 11 on page 66).
- "z" is 0 by default, but can increase depending on user actions, as documented for the formula to calculate the maximum number of address spaces (Figure 11 on page 66).

The definitions in Table 22 can limit the actual number of address spaces.

Table 22. Address space limits

Location	Limit	Affected resources
rsed.envvars	maximum.threadpool.process	Limits the number of RSE thread pools
IEASYMxx	MAXUSER	Limits the number of address spaces

Table 22. Address space limits (continued)

Location	Limit	Affected resources
ASCHPMxx	MAX	Limits the number of APPC initiators for TSO Commands service (APPC)

Process count

Table 23 lists the number of processes per address space that is used by Developer for System z. "u" In the "Address Spaces" column indicates that the amount must be multiplied by the number of concurrently active users using the function.

Table 23. Process count

Processes	Address spaces	Description	User ID
1	1	JES Job Monitor	STCJMON
3	1	Lock daemon	STCLOCK
3	1	RSE daemon	STCRSE
1	1	RSE APF authorized	STCRSE
2	(a)	RSE thread pool	STCRSE
2	(b)	ISPF Client Gateway (TSO Commands service and SCLMDT)	<userid>
1	1u	TSO Commands service (APPC)	<userid>
1	1u	CARMA (batch)	<userid>
1	1u	CARMA (crastart)	<userid>
1	1u	CARMA (ispf)	<userid>
1	3u	z/OS UNIX build (shell commands)	<userid>
1	1u	z/OS UNIX shell	<userid>
1	(c)	File Manager	<userid>
(5)	(u)	SCLM Developer Toolkit	<userid>

Note:

- (a) There is at least 1 RSE thread pool address space active. Refer to "Address space count" on page 65 to determine the actual number of RSE thread pool address spaces.
- RSE daemon and all RSE thread pools use the same user ID.
- (b) In normal situations, and when using the default configuration options, there is 1 ISPF Client Gateway active per user. The actual number can vary, as described in "Address space count" on page 65.
- (c) The File Manager listener uses a process per object that must be manipulated, for example a VSAM. This process stays active until Developer for System z signals that the object is no longer needed, for example by closing the VSAM.
- SCLMDT requires an ISPF Client Gateway address space. SCLMDT shares the address space with the TSO Commands service.
- (u) SCLMDT processes run in the ISPF Client Gateway address space, and thus do not have a value for the address space count.
- SCLMDT processes are temporary and end at task completion, but multiple processes can be active simultaneously for a single user. Table 23 lists the maximum number of concurrent SCLMDT processes.

- Most MVS data set-related actions use the TSO Commands service, which can be active in the ISPF Client Gateway or an APPC transaction, respectively.
- A z/OS UNIX build uses three processes in total, each running in their own address space.
- All listed processes stay active until the related address space ends, unless noted otherwise.

Use the formula in Figure 13 to estimate the maximum number of processes used by Developer for System z.

$$8 + 2*A + N*(x + y + z) + (10 + N*0.05)$$

Figure 13. Maximum number of processes

Where

- "8" equals the number of processes used by permanent active server address spaces.
- "A" represents the number of RSE thread pool address spaces.
- "N" represents the maximum number of concurrent users.
- "x" is one of the following values, depending on the selected configuration options.

X	SCLMDT	TSO by way of Client Gateway	TSO by way of APPC
1	No	No	Yes
2	No	Yes	No
7	Yes	Yes	No

- "y" is one of the following values, depending on the selected configuration options.

Y	
0	No CARMA
1	CARMA (batch)
1	CARMA (crastart)
4	CARMA (ispf)

- "z" is 0 by default, but can increase depending on user actions:
  - Add 1 when a z/OS UNIX shell is opened. This process stays active until the user logs off.
  - Add 3 when a z/OS UNIX build is performed. Note that the actual number may be higher, depending on the needs of the programs invoked. These processes end when the related build task completes.
  - Add 1 for each concurrent interaction with IBM File Manager. These processes end when the requested object is no longer needed.
- "10 + N*0.05" adds a buffer for temporary processes. The required buffer size might differ at your site.

Use the formula in Figure 14 on page 70 to estimate the maximum number of processes used by a Developer for System z client (not counting the undocumented

temporary processes).

$$(x + y + z) + 5*s$$

Figure 14. Number of processes per client

Where

- "x" depends on the selected configuration options and is documented for the formula to calculate the maximum number of processes (Figure 13 on page 69).
- "y" depends on the selected configuration options and is documented for the formula to calculate the maximum number of processes (Figure 13 on page 69).
- “z” is 0 by default, but can increase depending on user actions, as documented for the formula to calculate the maximum number of processes (Figure 13 on page 69).
- “s” is 1 when SCLM Developer Toolkit is used, or 0 otherwise.

The definitions in Table 24 can limit the actual number of processes.

Table 24. Process limits

Location	Limit	Affected resources
BPXPRMxx	MAXPROCSYS	Limits the total number of processes
BPXPRMxx	MAXPROCUSER	Limits the number of processes per z/OS UNIX UID

Note:

- RSE daemon and the RSE thread pools use the same user ID. Since RSE daemon starts a new thread pool whenever needed, the number of processes for this user ID can grow. So `MAXPROCUSER` must be set to accommodate this growth, which can be formulated as `"3 + 2*A"`.
- The `MAXPROCUSER` limit is per unique z/OS UNIX user ID (UID). Multiply the estimated per-user process count by the number of concurrently active clients if your users share the same UID.

Thread count

Table 25 on page 71 lists the number of threads used by selected Developer for System z functions. "u" In the "Threads" columns indicates that the amount must be multiplied by the number of concurrently active users using the function. The thread count is listed per process, as limits are set at this level.

- RSEDx: These threads are created in the RSE thread pool, which is shared by multiple clients. All threads ending up in the same thread pool must be added together to get the total count.
- Active: These threads are part of the process that actually does the requested function. Each process is a stand-alone unit, so there is no need to sum the thread counts, even if they are assigned to same user ID, unless noted otherwise.
- Bootstrap: Bootstrap processes are needed to start the actual process. Each has 1 thread, and there can be multiple consecutive bootstraps. There is no need to sum the thread counts.

Table 25. Thread count

Threads			User ID	Description
RSEDx	**Active**	**Bootstrap**		
-	3 + 1u	-	STCJMON	JES Job Monitor
-	10	2	STCLOCK	Lock daemon
-	11	2	STCRSE	RSE daemon
-	1	-	STCRSE	RSE APF authorized
10 (a) + 16u	-	1 (a)	STCRSE	RSE thread pool
-	4u (b)	1u (b)	<userid>	ISPF Client Gateway (TSO Commands service and SCLMDT)
-	2u	-	<userid>	TSO Commands service (APPC)
1u	2u	-	STCRSE and <userid>	CARMA (batch)
4u	2u	-	STCRSE and <userid>	CARMA (crastart)
5u	4u	3u	STCRSE and <userid>	CARMA (ispf)
-	1u (d)	2u	<userid>	z/OS UNIX build (shell commands)
6u	1u	-	STCRSE and <userid>	z/OS UNIX shell
-	2u (c)	-	<userid>	File Manager
-	(5)	-	<userid>	SCLM Developer Toolkit
1u	-	-	STCRSE	Timer for idle timeout

Note:

- (a) There is at least 1 RSE thread pool address space active. Refer to "Address space count" on page 65 to determine the actual number of RSE thread pool address spaces.
- (b) In normal situations, and when using the default configuration options, there is 1 ISPF Client Gateway active per user. The actual number can vary, as described in "Address space count" on page 65.
- (c) There is one user-specific process (with the listed thread count) per interaction with IBM File Manager. These processes end when the requested object is no longer needed.
- SCLMDT requires an ISPF Client Gateway address space. SCLMDT shares the address space with the TSO Commands service.
- Depending on the selected action, SCLMDT can use multiple single-thread processes that end at task completion. Table 25 lists the maximum number of concurrent SCLMDT threads.
- Most MVS data set-related actions use the TSO Commands service, which can be active in the ISPF Client Gateway or an APPC transaction, respectively.

- (d) A z/OS UNIX build invokes different build utilities, which might be multi-threaded. Table 25 on page 71 lists the minimum number of concurrent z/OS UNIX build threads.
- All listed threads stay active until the related process ends, unless noted otherwise.
- The normal thread count for RSE APF authorized code is 1. However, during startup, there are temporarily 13 or more simultaneous threads active.

Use the formula in Figure 15 to estimate the maximum number of threads used by a RSE thread pool. Use the formula in Figure 16 to estimate the maximum number of threads used by JES Job Monitor.

$$10 + N*(16 + x + y + z) + (20 + N*0.1)$$

Figure 15. Maximum number of RSE thread pool threads

$$3 + N$$

Figure 16. Maximum number of JES Job Monitor threads

Where

- "N" represents the maximum number of concurrent users in this thread pool or JES Job Monitor. The default settings aim for 60 users per thread pool.
- "x" is one of the following values, depending on the selected configuration options.

X	SCLMDT	TSO by way of Client Gateway	TSO by way of APPC	Timeout
0	No	No	Yes	No
0	No	Yes	No	No
0	Yes	Yes	No	No
1	No	No	Yes	Yes
1	No	Yes	No	Yes
1	Yes	Yes	No	Yes

- “y” is one of the following values, depending on the selected configuration options.

Y	
0	No CARMA
1	CARMA (batch)
4	CARMA (crastart)
5	CARMA (ispf)

- “z” is 0 by default, but can increase depending on user actions:
  - Add 6 when a z/OS UNIX shell is opened. These threads stay active until the user logs off.
- "20 + N*0.1" adds a buffer for temporary threads. The required buffer size might differ at your site.

The definitions in Table 26 can limit the actual number of threads in a process, which is mostly of importance for the RSE thread pools.

Table 26. Thread limits

Location	Limit	Affected resources
BPXPRMxx	MAXTHREADS	Limits the number of threads in a process.
BPXPRMxx	MAXTHREADTASKS	Limits the number of MVS tasks in a process.
BPXPRMxx	MAXASSIZE	Limits the address space size, and thus the storage available for thread related control blocks.
rsed.envvars	Xmx	Sets the maximum Java heap size. This storage is reserved and thus no longer available for thread related control blocks.
rsed.envvars	maximum.clients	Limits the number of clients (and thus their threads) in an RSE thread pool.
rsed.envvars	maximum.threads	Limits the number of client threads in a RSE thread pool.
FEJJCNFG	MAX_THREADS	Limits the number of threads in JES Job Monitor.

Note: The value for `maximum.threads` in `rsed.envvars` must be lower than the value for `MAXTHREADS` and `MAXTHREADTASKS` in `BPXPRMxx`.

Temporary resource usage

The resource usage documented in the previous sections is permanent for the life span of Developer for System z, or semi-permanent for certain user-specific tasks.

However, Developer for System z will temporarily use additional resources fore housekeeping tasks and to satisfy the following requests:

- Processing an audit file (`audit.action` directive in `rsed.envvars`) uses one additional thread, one additional process, and possibly (if `audit.action.id` is set) one additional address space.
- Operator command IVP PASSTICKET will use two additional threads.
- Operator command IVP DAEMON will use one additional thread, one additional process, and one additional address space.
- Operator command IVP ISPF will use one additional thread, one additional process, and one additional address space, plus the resources used by ISPF Client Gateway.

Storage usage

RSE is a Java application, which implies that storage (memory) usage planning for Developer for System z must take two storage allocation limits into consideration, Java heap size and Address Space size.

Java heap size limit

Java offers many services to ease coding efforts for Java applications. One of these services is storage management.

Java's storage management allocates large blocks of storage and uses these for storage requests by the application. This storage managed by Java is called the Java heap. Periodic garbage collection (defragmentation) reclaims unused space in the heap and reduces its size.

The maximum Java heap size is defined in `rsed.envvars` with the `Xmx` directive. If this directive is not specified, Java uses a default size of 512 MB (64 MB for Java 5.0).

Each RSE thread pool (which services the client actions) is a separate Java application, and thus has a personal Java heap. Note that all thread pools use the same `rsed.envvars` configuration file, and thus have the same Java heap size limit.

The thread pool's usage of the Java heap depends heavily on the actions done by the connected clients. Regular monitoring of the heap usage is required to set the optimal heap size limit. Use the **modify display process** operator command to monitor the Java heap usage by RSE thread pools.

Address space size limit

All z/OS applications, including Java applications, are active within an address space, and are thus bound by address space size limitations.

The desired address space size is specified during startup, for example with the REGION parameter in JCL. However, system settings can limit the actual address space size. Refer to "Address Space size" on page 151 to learn more about these limits.

- `MAXASSIZE` in `SYS1.PARMLIB(BPXPRMxx)`
- `ASSIZEMAX` in the OMVS segment of the user ID assigned to the started task
- system exits `IEFUSI` and `IEALIMIT`

RSE thread pools inherit the address space size limits from RSE daemon. The address space size must be sufficient to house the Java heap, Java itself, common storage areas, and all control blocks the system creates to support the thread pool activity, such as a TCB (Task Control Block) per thread. Note that some of this storage usage is below the 16 MB line.

You should monitor the actual address space size before changing any settings that influence it, like changing the size of the Java heap or the amount of users supported by a single thread pool. Use your regular system monitoring software to track the actual storage usage by Developer for system z. If you do not have a dedicated monitoring tool, then basic information can be gathered with tools like the SDSF DA view or TASID (an as-is system information tool available via the ISPF "Support and downloads" webpage).

Size estimate guidelines

As stated before, the actual storage usage by Developer for system z is heavily influenced by user activity. Some actions use a fixed amount of storage (for example, logon), while others are variable (for example, listing data sets with a specified high-level qualifier).

- Use a 2 GB address space for RSE to allow room for the Java heap and all the system control blocks.
- The sample `rsed.envvars` setup aims for 60 users per thread pool.
  - `maximum.clients=60`
  - `maximum.threads=1000` (10+16*60 = 970, so 1000 allows for 61 clients)
- The sample `rsed.envvars` setup lets the Java heap grow up to 256 MB. This allows for 60 clients using an average of 4 MB per client (60*4 = 240).

Note that RSE displays the current Java heap and address space size limit during startup in console message FEK004I.

Use either of the following scenarios if monitoring shows that the current Java heap size is insufficient for the actual workload:

- Increase the maximum Java heap size with the `Xmx` directive in `rsed.envvars`. Before doing so, ensure that there is room in the address space for the size increase.
- Decrease the maximum number of clients per thread pool with the `maximum.clients` directive in `rsed.envvars`. RSE will still support the same number of clients, but the clients will be distributed among more thread pools.

Sample storage usage analysis

The displays in the following figures show some sample resource usage numbers for a default Developer for system z setup with these modifications.

- `single.logon` is disabled to stop RSE from creating at least 2 thread pool address spaces
- The maximum Java heap size is set to 10 MB, as a small maximum will result in a bigger percentile usage and the heap size limits will be reached sooner.

```
Max Heap Size=10MB and private AS Size=1,959MB

startup

 BPXM023I (STCRSE)
 ProcessId(268     ) Memory Usage(7%) Clients(0)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              0.01      2740          72
 LOCKD             1.60     28.7M        14183
 RSED              4.47     32.8M        15910
 RSED8             1.15     27.4M        12612

logon 1

 BPXM023I (STCRSE)
 ProcessId(268     ) Memory Usage(13%) Clients(1)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              0.01      2864          81
 LOCKD             1.64     28.8M        14259
 RSED              4.55     32.8M        15980
 RSED8             3.72     55.9M        24128

logon 2

 BPXM023I (STCRSE)
 ProcessId(268     ) Memory Usage(23%) Clients(2)

 Jobname     Cpu time      Storage     EXCP
--------   -----------    -------  ----------
 JMON              0.02      2944          86
 LOCKD             1.66     28.9M        14268
 RSED              4.58     32.9M        16027
 RSED8             4.20     57.8M        25205

logon 3

 BPXM023I (STCRSE)
 ProcessId(268     ) Memory Usage(37%) Clients(3)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              0.02      3020          91
 LOCKD             1.67     29.0M        14277
 RSED              4.60     32.9M        16076
 RSED8             4.51     59.6M        26327

logon 4

 BPXM023I (STCRSE)
 ProcessId(268     ) Memory Usage(41%) Clients(4)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              0.02      3108          96
 LOCKD             1.68     29.0M        14286
 RSED              4.61     32.9M        16125
 RSED8             4.77     62.3M        27404
```

Figure 17. Resource usage with 5 logons

```
logon 5

 BPXM023I (STCRSE)
 ProcessId(268     ) Memory Usage(41%) Clients(4)
 ProcessId(33554706) Memory Usage(13%) Clients(1)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              0.03       3184         101
 LOCKD             1.69      29.1M       14295
 RSED              4.64      32.9M       16229
 RSED8             4.78      62.4M       27413
 RSED9             4.60      56.6M       24065
```

Figure 18. Resource usage with 5 logons (continued)

Figure 17 on page 76 and Figure 18 show a scenario where 5 clients log on to an RSE daemon with a 10 MB Java heap.

- A thread pool (RSED8) is in a dormant state upon startup, using about 27 MB, of which 0.7 MB is in the Java heap (7% of 10 MB).
- The thread pool becomes active when the first client connects, using another 27 MB plus 2 MB for each client that connects.
- Part of this 2MB per connection will be in the Java heap, as the increase in heap usage shows.
- However, there is no real pattern in the heap usage, because it depends on Java mechanisms that estimate the required storage and allocate more than needed. Intermittent garbage collection frees up storage, making trends even harder to detect.
- Internal mechanisms that limit the number of connections per thread pool to ensure sufficient heap size for the active threads result in the fifth connection being created in a new thread pool (RSED9). These internal safety nets are normally not invoked when using a properly configured setup, because other limits would be reached first (most likely the `maximum.clients` one in `rsed.envvars`).

```
Max Heap Size=10MB and private AS Size=1,959MB

startup

 BPXM023I (STCRSE)
 ProcessId(212     ) Memory Usage(7%) Clients(0)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              0.01      2736          71
 LOCKD             1.73     30.5M        14179
 RSED              4.35     32.9M        15117
 RSED8             1.43     27.4M        12609

logon

 BPXM023I (STCRSE)
 ProcessId(212     ) Memory Usage(13%) Clients(1)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              0.01      2864          80
 LOCKD             1.76     30.6M        14255
 RSED              4.48     33.0M        15187
 RSED8             3.53     53.9M        24125

expand large MVS tree (195 data sets)
 BPXM023I (STCRSE)
 ProcessId(212     ) Memory Usage(13%) Clients(1)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              0.01      2864          80
 LOCKD             1.78     30.6M        14255
 RSED              4.58     33.1M        16094
 RSED8             4.28     56.1M        24740

expand small PDS (21 members)
 BPXM023I (STCRSE)
 ProcessId(212     ) Memory Usage(13%) Clients(1)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 IBMUSER2          0.22      2644         870
 JMON              0.01      2864          80
 LOCKD             1.78     30.6M        14255
 RSED              4.61     33.1M        16108
 RSED8             4.40     56.2M        24937

open medium sized member (86 lines)

 BPXM023I (STCRSE)
 ProcessId(212     ) Memory Usage(13%) Clients(1)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 IBMUSER2          0.22      2644         870
 JMON              0.01      2864          80
 RSED              4.61     33.1M        16108
 RSED8             8.12     62.7M        27044
```

Figure 19. Resource usage while editing a PDS member

Figure 19 shows a scenario where 1 client logs on to an RSE daemon with a 10 MB Java heap and edits a PDS member.

- The catalog search that results in 195 data set names used about 2MB of storage, all due to system activity, because the Java heap usage does not increase.
- Opening a 21-member PDS uses hardly any memory in the thread pool, but the display shows that TSO Commands service was invoked. There is a new address space active (IBMUSER2), which uses the region size assigned to this user ID in TSO. This address space stays active for a specified amount of time so it can be reused for future requests by the TSO Commands service.
- Opening a member shows similar numbers as expanding a high-level qualifier. The Java heap usage stays the same, but there is a 6.5 MB storage increase due to system activity.

z/OS UNIX file system space usage

Most of the Developer for System z related data that is not written to a DD statement ends up in a z/OS UNIX file. The system programmer has control over which data is written and where it goes. However, there is no control over the amount of data written.

The data can be grouped in the following categories:

- Problem analysis (log and system dump files), for which many details are documented in Chapter 11, "Troubleshooting configuration problems," on page 137
- Auditing, as documented in "Audit logging" on page 19
- Push-to-client metadata, as documented in "Push-to-client metadata" on page 100.
- Temporary data

As documented in Chapter 11, "Troubleshooting configuration problems," on page 137, Developer for System z writes the RSE-related host logs to the following z/OS UNIX directories:

- `/var/rdz/logs` for the RSE started task logs
- `/var/rdz/logs/$LOGNAME` for user logs

By default, only error and warning messages are written to the logs. So if all goes as planned, these directories should hold only empty or nearly-empty files (not counting audit logs).

You can enable logging of informational messages, preferably under direction of the IBM support center, which increases the size of log files noticeably.

```
startup

$ ls -l /var/rdz/logs
total 144
-rw-rw-rw-   1 STCRSE   STCGRP     33642 Jul 10 12:10 rsedaemon.log
-rw-rw-rw-   1 STCRSE   STCGRP      1442 Jul 10 12:10 rseserver.log

logon

$ ls -l /var/rdz/logs
total 144
drwxrwxrwx   3 IBMUSER  SYS1        8192 Jul 10 12:11 IBMUSER
-rw-rw-rw-   1 STCRSE   STCGRP     36655 Jul 10 12:11 rsedaemon.log
-rw-rw-rw-   1 STCRSE   STCGRP      1893 Jul 10 12:11 rseserver.log
$ ls -l /var/rdz/logs/IBMUSER
total 160
-rw-rw-rw-   1 IBMUSER  SYS1        3459 Jul 10 12:11 ffs.log
-rw-rw-rw-   1 IBMUSER  SYS1           0 Jul 10 12:11 ffsget.log
-rw-rw-rw-   1 IBMUSER  SYS1           0 Jul 10 12:11 ffsput.log
-rw-rw-rw-   1 IBMUSER  SYS1         303 Jul 10 12:11 lock.log
-rw-rw-rw-   1 IBMUSER  SYS1         126 Jul 10 12:11 rmt_classloader_cache.jar
-rw-rw-rw-   1 IBMUSER  SYS1        7266 Jul 10 12:11 rsecomm.log
-rw-rw-rw-   1 IBMUSER  SYS1           0 Jul 10 12:11 stderr.log
-rw-rw-rw-   1 IBMUSER  SYS1           0 Jul 10 12:11 stdout.log

logoff
$ ls -l /var/rdz/logs
total 80
drwxrwxrwx   3 IBMUSER  SYS1        8192 Jul 10 12:11 IBMUSER
-rw-rw-rw-   1 STCRSE   STCGRP     36655 Jul 10 12:11 rsedaemon.log
-rw-rw-rw-   1 STCRSE   STCGRP      2208 Jul 10 12:11 rseserver.log
$ ls -l /var/rdz/logs/IBMUSER
total 296
-rw-rw-rw-   1 IBMUSER  SYS1        6393 Jul 10 12:11 ffs.log
-rw-rw-rw-   1 IBMUSER  SYS1           0 Jul 10 12:11 ffsget.log
-rw-rw-rw-   1 IBMUSER  SYS1           0 Jul 10 12:11 ffsput.log
-rw-rw-rw-   1 IBMUSER  SYS1         609 Jul 10 12:11 lock.log
-rw-rw-rw-   1 IBMUSER  SYS1         126 Jul 10 12:11 rmt_classloader_cache.jar
-rw-rw-rw-   1 IBMUSER  SYS1       45157 Jul 10 12:11 rsecomm.log
-rw-rw-rw-   1 IBMUSER  SYS1           0 Jul 10 12:11 stderr.log
-rw-rw-rw-   1 IBMUSER  SYS1         176 Jul 10 12:11 stdout.log

stop

$ ls -l /var/rdz/logs
total 80
drwxrwxrwx   3 IBMUSER  SYS1        8192 Jul 10 12:11 IBMUSER
-rw-rw-rw-   1 STCRSE   STCGRP     36655 Jul 10 12:11 rsedaemon.log
-rw-rw-rw-   1 STCRSE   STCGRP      2490 Jul 10 12:12 rseserver.log
```

Figure 20. z/OS UNIX file system space usage

Figure 20 shows the minimal z/OS UNIX file system space usage when using debug level 2 (informational messages).

- The started task logs use 34 KB after startup and grow slowly when users log on, log off, or operator commands are issued.
- A client log directory uses 11 KB after logon and grows at a steady pace when the user starts working (not shown in the sample).
- Logoff adds another 40 KB to the user logs, bringing them to 51 KB.

Except for audit logs, log files are overwritten on every restart (for the RSE started task) or logon (for a client), keeping the total size in check. The `keep.last.log` directive in `rsed.envvars` changes this slightly, as it can instruct RSE to keep a copy of the previous logs. Older copies are always removed.

A warning message is sent to the console when the file system holding the audit log files is running low on free space and auditing is active. This console message (FEK103E) is repeated regularly until the low space issue is resolved. Refer to "Console messages" in the *Host Configuration Guide* (SC23-7658) for a list of console messages generated by RSE.

The definitions in Table 27 control which data is written to the log directories, and where the directories are located.

Table 27. Log output directives

Location	Directive	Function
resecomm.properties	debug_level	Set the default log detail level.
rsed.envvars	keep.last.log	Keep a copy of the previous logs before startup/logon.
rsed.envvars	enable.audit.log	Keep an audit trace of client actions.
rsed.envvars	enable.standard.log	Write the stdout and stderr streams of the thread pool (or pools) to a log file.
rsed.envvars	DSTORE_TRACING_ON	Enable DataStore action logging.
rsed.envvars	DSTORE_MEMLOGGING_ON	Enable DataStore memory usage logging.
Operator command	modify rsecommlog <level>	Dynamically change the log detail level of rsecomm.log
Operator command	modify rsedaemonlog <level>	Dynamically change the log detail level of rsedaemon.log
Operator command	modify rseserverlog <level>	Dynamically change the log detail level of rseserver.log
Operator command	modify rsestandardlog {on|off}	Dynamically change the updating of std*.*.log
rsed.envvars	daemon.log	Home path for RSE started task and audit logs.
rsed.envvars	user.log	Home path for user logs.
rsed.envvars	_CMDSERV_WORK_HOME	Home path for ISPF Client Gateway logs
rsed.envvars	TMPDIR	Directory for IVP logs
rsed.envvars	_CEE_DMPTARG	Directory for Java dumps

Developer for System z, together with requisite software such as the ISPF Client Gateway, also writes temporary data to `/tmp` and `/var/rdz/WORKAREA`. The amount of data written here as a result of user actions is unpredictable, so you should have ample free space in the file systems holding these directories.

Developer for System z always tries to clean up these temporary files, but manual cleanup, as documented in "(Optional) WORKAREA and /tmp cleanup" in the *Host Configuration Guide* (SC23-7658), can be performed at virtually any time.

The definitions in Table 28 on page 82 control where temporary data directories are located.

Table 28. Temporary output directives

Location	Directive	Function
rsed.envvars	_CMDSERV_WORK_HOME	Home path for temporary data.
rsed.envvars	TMPDIR	Directory for temporary data.

Key resource definitions

/etc/rdz/rsed.envvars

The environment variables defined in `rsed.envvars` are used by RSE, Java, and z/OS UNIX. The sample file that comes with Developer for System z is targeted at small to medium sized installations that do not require the optional components of Developer for System z. "rsed.envvars, RSE configuration file" in the *Host Configuration Guide* (SC23-7658) describes each variable that is defined in the sample file, where the following variables require special attention:

`_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Xms128m -Xmx256m"`
Set initial (`Xms`) and maximum (`Xmx`) heap size. The defaults are `128M` and `256M` respectively. Change to enforce the desired heap size values. If this directive is commented out, the Java default values will be used, which are `4M` and `512M` respectively (`1M` and `64M` for Java 5.0).

`#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dmaximum.clients=60"`
Maximum amount of clients serviced by one thread pool. The default is `60`. Uncomment and customize to limit the number of clients per thread pool. Note that other limits may prevent RSE from reaching this limit.

`#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dmaximum.threads=1000"`
Maximum amount of active threads in one thread pool to allow new clients. The default is `1000`. Uncomment and customize to limit the number of clients per thread pool based upon the number of threads in use. Note that each client connection uses multiple threads (16 or more) and that other limits may prevent RSE from reaching this limit.

Note: This value must be lower than the setting for `MAXTHREADS` and `MAXTHREADTASKS` in `SYS1.PARMLIB(BPXPRMxx)`.

`#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dminimum.threadpool.process=1"`
The minimum number of active thread pools. The default is 1. Uncomment and customize to start at least the listed number of thread pool processes. Thread pool processes are used for load balancing the RSE server threads. More new processes are started when they are needed. Starting the new processes up front helps prevent connection delays but uses more resources during idle times.

Note: If the `single.logon` directive is active, then there will be at least 2 thread pools started, even if `minimum.threadpool.process` is set to 1. The default setting for `single.logon` in `rsed.envvars` is active.

`#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dmaximum.threadpool.process=100"`
The maximum number of active thread pools. The default is `100`. Uncomment and customize to limit the number of thread pool processes. Thread pool processes are used for load balancing the RSE server threads, so limiting them will limit the amount of active client connections.

SYS1.PARMLIB(BPXPRMxx)

RSE is a Java application, which means that it is active in the z/OS UNIX environment. This promotes BPXPRMxx to become a crucial parmlib member, as it contains the parameters that control the z/OS UNIX environment and file systems. BPXPRMxx is described in the *MVS Initialization and Tuning Reference* (SA22-7592). The following directives are known to impact Developer for System z:

MAXPROCSYS(nnnnn)
Specifies the maximum number of processes that the system allows.

Value Range: nnnnn is a decimal value from 5 to 32767.
Default: 900

MAXPROCUSER(nnnnn)
Specifies the maximum number of processes that a single z/OS UNIX user ID can have concurrently active, regardless of how the processes were created.

Value Range: nnnnn is a decimal value from 3 to 32767.
Default: 25

Note:

- All RSE processes use the same z/OS UNIX user ID (that of the user who is assigned to RSE daemon), because all clients run as threads within the RSE processes.
- This value can also be set with the PROCUSERMAX variable in the OMVS security profile segment of the user assigned to the RSED started task.

MAXTHREADS(nnnnnn)
Specifies the maximum number of pthread_created threads, including running, queued, and exited but undetached, that a single process can have concurrently active. Specifying a value of 0 prevents applications from using pthread_create.

Value Range: nnnnnn is a decimal value from 0 to 100000.
Default: 200

Note:

- Each client uses at least 16 threads within the RSE thread pool process, and multiple clients are active within the process.
- This value can also be set with the THREADSMAX variable in the OMVS security profile segment of the user assigned to the RSED started task. When set, the THREADSMAX value is used for both MAXTHREADS and MAXTHREADTASKS.

MAXTHREADTASKS(nnnnn)
Specifies the maximum number of MVS tasks that a single process can have concurrently active for pthread_created threads.

Value Range: nnnnn is a decimal value from 0 to 32768.
Default: 1000

Note:

- Each active thread has an MVS task (TCB, Task Control Block).
- Each concurrent MVS task requires additional storage, some of which must be below the 16MB line.

- Each client uses at least 16 threads within the RSE thread pool process, and multiple clients are active within the process.
- This value can also be set with the THREADSMAX variable in the OMVS security profile segment of the user assigned to the RSED started task. When set, the THREADSMAX value is used for both MAXTHREADS and MAXTHREADTASKS.

MAXUIDS(nnnnn)
Specifies the maximum number of z/OS UNIX user IDs (UIDs) that can operate concurrently.

Value Range: nnnnn is a decimal value from 1 to 32767.
Default: 200

MAXASSIZE(nnnnn)
Specifies the RLIMIT_AS resource values that will be established as the initial values for new processes. RLIMIT_AS indicates the address space region size.

Value Range: nnnnn is a decimal value from 10485760 (10 Megabytes) to 2147483647 (2 Gigabytes).
Default: 209715200 (200 Megabytes)

Note:
- This value should be set to 2G.
- This value can also be set with the ASSIZEMAX variable in the OMVS security profile segment of the user assigned to the RSED started task.

MAXFILEPROC(nnnnnn)
Specifies the maximum number of descriptors for files, sockets, directories, and any other file system objects that a single process can have concurrently active or allocated.

Value Range: nnnnnn is a decimal value from 3 to 524287.
Default: 64000

Note:
- A thread pool has all his client threads in a single process.
- This value can also be set with the FILEPROCMAX variable in the OMVS security profile segment of the user assigned to the RSED started task.

MAXMMAPAREA(nnnnn)
Specifies the maximum amount of data space storage space (in pages) that can be allocated for memory mappings of z/OS UNIX files. Storage is not allocated until the memory mapping is active.

Value Range: nnnnn is a decimal value from 1 to 16777216.
Default: 40960

Note: This value can also be set with the MMAPAREAMAX variable in the OMVS security profile segment of the user assigned to the RSED started task.

Use the **SETOMVS** or **SET OMVS** operator command to dynamically (until next IPL) increase or decrease the value of any of the previous BPXPRMxx variables. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs. Refer to *MVS System Commands* (SA22-7627) for more information on these operator commands.

The following definitions are sub-parameters of the NETWORK statement.

MAXSOCKETS(nnnnnnnn)
Specifies the maximum number of sockets supported by this file system for this address family. This is an optional parameter.

Value Range: nnnnnnnn is a decimal value from 0 to 16777215.
Default: 100

INADDRANYCOUNT(nnnn)
Specifies the number of ports that the system reserves for use with PORT 0, INADDR_ANY binds, starting with the port number specified in the INADDRANYPORT parameter. This value is only needed for CINET (multiple TCP/IP stacks).

Value Range: nnnn is a decimal value from 1 to 4000.
Default: If neither INADDRANYPORT or INADDRANYCOUNT is specified, the default for INADDRANYCOUNT is 1000. Otherwise, no ports are reserved (0).

Various resource definitions

EXEC card in the server JCL

The following definitions are recommended to be added to the EXEC card in the JCL of the Developer for System z servers.

REGION=0M
`REGION=0M` is recommended for the RSE daemon and JES Job Monitor started tasks, RSED and JMON respectively. By doing so, the address space size is limited only by the available private storage, or by the IEFUSI or IEALIMIT system exits. Note that IBM strongly recommends not to use these exits for z/OS UNIX address spaces, like RSE daemon.

TIME=NOLIMIT
`TIME=NOLIMIT` is recommended to be used for all Developer for System z servers. This because the CPU time of all Developer for System z clients accumulates in the server address spaces.

FEK.#CUST.PARMLIB(FEJJCNFG)

The environment variables defined in `FEJJCNFG` are used by JES Job Monitor. The sample file that comes with Developer for System z is targeted at small to medium sized installations. "FEJJCNFG, JES Job Monitor configuration file" in the *Host Configuration Guide* (SC23-7658) describes each variable that is defined in the sample file, where the following variables require special attention:

MAX_THREADS
Maximum number of users that can be using one JES Job Monitor at a time. The default is `200`. The maximum value is `2147483647`. Increasing this number may require you to increase the size of the JES Job Monitor address space.

SYS1.PARMLIB(IEASYSxx)

IEASYSxx holds system parameters and is described in the *MVS Initialization and Tuning Reference* (SA22-7592). The following directives are known to impact Developer for System z:

MAXUSER=nnnnn
This parameter specifies a value that, under most conditions, the system uses to limit the number of jobs and started tasks that can run concurrently during a given IPL.

Value Range: nnnnn is a decimal value from 0-32767. Note that the sum of the values specified for the MAXUSER, RSVSTRT, and RSVNONR system parameters cannot exceed 32767.
Default Value: 255

SYS1.PARMLIB(IVTPRMxx)

IVTPRMxx sets parameters for the Communication Storage Manager (CSM), and is described in the *MVS Initialization and Tuning Reference* (SA22-7592). The following directives are known to impact Developer for System z:

FIXED MAX(maxfix)
Defines the maximum amount of storage dedicated to fixed CSM buffers.

Value Range: maxfix is a value from 1024K to 2048M.
Default: 100M

ECSA MAX(maxecsa)
Defines the maximum amount of storage dedicated to ECSA CSM buffers.

Value Range: maxecsa is a value from 1024K to 2048M.
Default: 100M

SYS1.PARMLIB(ASCHPMxx)

The ASCHPMxx parmlib member contains scheduling information for the ASCH transaction scheduler, and is described in the *MVS Initialization and Tuning Reference* (SA22-7592). The following directives are known to impact Developer for System z:

MAX(nnnnn)
An optional parameter of the CLASSADD definition that specifies the maximum number of APPC transaction initiators that are allowed for a particular class of transaction initiators. After this limit is reached, no new address spaces are created and incoming requests are queued to wait until existing initiator address spaces become available. The value should not exceed the maximum number of address spaces allowed by your installation, and you should be aware of competing products on the system that will also require address spaces.

Value Range: nnnnn is a decimal value from 1 to 64000.
Default: 1

Note: If you use APPC to start the TSO Commands service, then the transaction class used must have enough transaction initiators to allow one initiator for each concurrent user of Developer for System z.

Monitoring

Since user workloads can change the need for system resources, the system should be monitored regularly to measure resource usage so that Rational Developer for System z and system configurations can be adjusted in response to user requirements. The following commands can be used to aid in this monitoring process.

Monitoring RSE

RSE thread pools are the focal point for user activity in Developer for System z, and thus require monitoring for optimal use. RSE daemon can be queried for information that cannot be gathered with regular system monitoring tools.

- Use your regular system monitoring tools, such as RMF™, to gather address space-specific data such as used real storage and CPU-time. If you do not have a dedicated monitoring tool, then basic information can be gathered with tools like the SDSF DA view or TASID (an as-is system information tool available via the ISPF "Support and downloads" webpage).
- During startup, the RSE daemon reports the available address space size and Java heap size with console message FEK004I.

  ```
  FEK004I RseDaemon: Max Heap Size=65MB and private AS Size=1,959MB
  ```

- The **MODIFY RSED,APPL=DISPLAY PROCESS** operator command displays the RSE thread pool processes. The "Memory Usage" field shows how much of the defined Java heap is actually used. Refer to "Operator commands" in the *Host Configuration Guide* (SC23-7658) for more information on this command.

  ```
  f rsed,appl=d p
  BPXM023I (STCRSE)
  ProcessId(16777456) Memory Usage(33%) Clients(4) Order(1)
  ```

  More information is provided when the DETAIL option of the **DISPLAY PROCESS** modify command is used:

  ```
  f rsed,appl=d p,detail
  BPXM023I (STCRSE)
  ProcessId(33555087) ASId(002E) JobName(RSED8) Order(1)
   PROCESS LIMITS:    CURRENT  HIGHWATER      LIMIT
    JAVA HEAP USAGE(%)     10         56        100
    CLIENTS                 0         25         60
    MAXFILEPROC            83        103      64000
    MAXPROCUSER            97         99        200
    MAXTHREADS              9         14       1500
    MAXTHREADTASKS          9         14       1500
  ```

- When an RSE thread pool process ends, it displays detailed resource usage statistics, as if the **DISPLAY PROCESS,DETAIL** modify command was issued for just that RSE thread pool process. The high-water mark shows the maximum concurrent resource usage for the life of the RSE thread pool process, allowing a system tuner to determine if resources assigned to RSE are over-allocated or under-allocated.

Monitoring z/OS UNIX

Most z/OS UNIX limits that are of interest for Developer for System z can be displayed using operator commands. Some commands even show the current usage and the high-water mark for a specific limit. Refer to *MVS System Commands* (SA22-7627) for more information on these commands.

- The `LIMMSG(ALL)` directive in `SYS1.PARMLIB(BPXPRMxx)` tells z/OS UNIX to display console messages (`BPXI040I`) when any of the parmlib limits is about to be reached. The default value for `LIMMSG` is `NONE`, which disables the function. Use operator command **SETOMVS LIMMSG=ALL** to dynamically activate this function (until next IPL). Refer to *MVS Initialization and Tuning Reference* (SA22-7592) for more information on this directive.
- The **DISPLAY OMVS,OPTIONS** operator command displays the current values of z/OS UNIX directives that can be set dynamically.

  ```
  d omvs,o
  BPXO043I 13.10.16 DISPLAY OMVS 066
  OMVS     000D ETC/INIT WAIT   OMVS=(M7)
  CURRENT UNIX CONFIGURATION SETTINGS:
  ```

```
MAXPROCSYS      =        256    MAXPROCUSER     =         16
MAXFILEPROC     =        256    MAXFILESIZE     = NOLIMIT
MAXCPUTIME      =       1000    MAXUIDS         =        200
MAXPTYS         =        256
MAXMMAPAREA     =        256    MAXASSIZE       =  209715200
MAXTHREADS      =        200    MAXTHREADTASKS  =       1000
MAXCORESIZE     =    4194304    MAXSHAREPAGES   =       4096
IPCMSGQBYTES    = 2147483647    IPCMSGQMNUM     =      10000
IPCMSGNIDS      =        500    IPCSEMNIDS      =        500
IPCSEMNOPS      =         25    IPCSEMNSEMS     =       1000
IPCSHMMPAGES    =      25600    IPCSHMNIDS      =        500
IPCSHMNSEGS     =        500    IPCSHMSPAGES    =     262144
SUPERUSER       = BPXROOT       FORKCOPY        = COW
STEPLIBLIST     =
USERIDALIASTABLE=
SERV_LINKLIB    = POSIX.DYNSERV.LOADLIB   BPXLK1
SERV_LPALIB     = POSIX.DYNSERV.LOADLIB   BPXLK1
PRIORITYPG VALUES: NONE
PRIORITYGOAL VALUES: NONE
MAXQUEUEDSIGS   =       1000    SHRLIBRGNSIZE   =   67108864
SHRLIBMAXPAGES  =       4096    VERSION         = /
SYSCALL COUNTS  = NO            TTYGROUP        = TTY
SYSPLEX         = NO            BRLM SERVER     = N/A
LIMMSG          = NONE          AUTOCVT         = OFF
RESOLVER PROC   = DEFAULT
AUTHPGMLIST     = NONE
SWA             = BELOW
```

- The **DISPLAY OMVS,LIMITS** operator command displays information about current z/OS UNIX System Services parmlib limits, their high-water marks, and current system usage.

```
d omvs,l
BPXO051I 14.05.52 DISPLAY OMVS 904
OMVS     0042 ACTIVE          OMVS=(69)
SYSTEM WIDE LIMITS:         LIMMSG=SYSTEM
                  CURRENT  HIGHWATER     SYSTEM
                    USAGE      USAGE      LIMIT
MAXPROCSYS              1          4        256
MAXUIDS                 0          0        200
MAXPTYS                 0          0        256
MAXMMAPAREA             0          0        256
MAXSHAREPAGES           0         10       4096
IPCMSGNIDS              0          0        500
IPCSEMNIDS              0          0        500
IPCSHMNIDS              0          0        500
IPCSHMSPAGES            0          0     262144 *
IPCMSGQBYTES          ---          0     262144
IPCMSGQMNUM           ---          0      10000
IPCSHMMPAGES          ---          0        256
SHRLIBRGNSIZE           0          0   67108864
SHRLIBMAXPAGES          0          0       4096
```

  The command displays high-water marks and current usage for an individual process when the `PID=processid` keyword is also specified.

```
d,omvs,l,pid=16777456
BPXO051I 14.06.28 DISPLAY OMVS 645
OMVS     000E ACTIVE             OMVS=(76)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
STCRSE   RSED8    007E   16777456   67109106 HF---- 20.00.56   113.914
 LATCHWAITPID=         0 CMD=java -Ddaemon.log=/var/rdz/logs -
PROCESS LIMITS:        LIMMSG=NONE
                  CURRENT  HIGHWATER    PROCESS
                    USAGE      USAGE      LIMIT

MAXFILEPROC            83        103        256
MAXFILESIZE           ---        ---    NOLIMIT
MAXPROCUSER            97         99        200
```

```
MAXQUEUEDSIGS           0          1       1000
MAXTHREADS              9         14        200
MAXTHREADTASKS          9         14       1000
IPCSHMNSEGS             0          0        500
MAXCORESIZE           ---        ---    4194304
MAXMEMLIMIT             0          0     16383P
```

- The **DISPLAY OMVS,PFS** operator command displays information about each physical file system that is currently part of the z/OS UNIX configuration, which includes the TCP/IP stacks.

```
d omvs,p
BPXO046I 14.35.38 DISPLAY OMVS 092
OMVS     000E ACTIVE          OMVS=(33)
PFS CONFIGURATION INFORMATION
 PFS TYPE     DESCRIPTION            ENTRY      MAXSOCK   OPNSOCK   HIGHUSED
  TCP          SOCKETS AF_INET        EZBPFINI    50000       244       8146
  UDS          SOCKETS AF_UNIX        BPXTUINT       64         6         10
  ZFS          LOCAL FILE SYSTEM      IOEFSCM
               14:32.00 RECYCLING
  HFS          LOCAL FILE SYSTEM      GFUAINIT
  BPXFTCLN     CLEANUP DAEMON        BPXFTCLN
  BPXFTSYN     SYNC DAEMON           BPXFTSYN
  BPXFPINT     PIPE                  BPXFPINT
  BPXFCSIN     CHAR SPECIAL          BPXFCSIN
  NFS          REMOTE FILE SYSTEM    GFSCINIT
 PFS NAME     DESCRIPTION           ENTRY     STATUS   FLAGS
  TCP41        SOCKETS               EZBPFINI  ACT      CD
  TCP42        SOCKETS               EZBPFINI  ACT
  TCP43        SOCKETS               EZBPFINI  INACT    SD
  TCP44        SOCKETS               EZBPFINI  INACT
PFS PARM INFORMATION
 HFS         SYNCDEFAULT(60) FIXED(50) VIRTUAL(100)
             CURRENT VALUES: FIXED(55) VIRTUAL(100)
 NFS         biod(6)
```

- The **DISPLAY OMVS,PID=processid** operator command displays the thread information for a specific process.

```
d omvs,pid=16777456
BPXO040I 15.30.01 DISPLAY OMVS 637
OMVS     000E ACTIVE             OMVS=(76)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
STCRSE   RSED8    007E   16777456   67109106 HF---- 20.00.56   113.914
 LATCHWAITPID=         0 CMD=java -Ddaemon.log=/var/rdz/logs -
THREAD_ID        TCB@     PRI_JOB  USERNAME   ACC_TIME SC  STATE
0E08A000000000000 005E6DF0 OMVS                   .927 RCV  FU
0E08F000000000001 005E6C58                        .001 PTX JYNV
0E09300000000002 005E6AC0                        7.368 PTX JYNV
0E0CB000000000008 005C2CF0 OMVS                  1.872 SEL JFNV
0E192000000003CE 005A0B70 OMVS     IBMUSER      14.088 POL JFNV
0E18D000000003CF 005A1938          IBMUSER        .581 SND JYNV
```

Monitoring the network

When supporting a large number of clients connecting to the host, then not only Developer for System z, but also your network infrastructure must be able to handle the workload. Network management is a broad and well documented subject that falls out of the scope of Developer for System z documentation. Therefore, only the following pointers are provided.

- The **DISPLAY NET,CSM** operator command allows you to monitor the use of storage managed by the communications storage manager (CSM). You can use this command to determine how much CSM storage is in use for ECSA and data space storage pools, as documented in *Communications Server SNA Operations* (SC31-8779).

Monitoring z/OS UNIX file systems

Developer for System z uses z/OS UNIX file systems to store various types of data, such as logs and temporary files. Use the z/OS UNIX **df** command to see how many file descriptors are still available and how much free space is left before the next extent of the underlying HFS or zFS data set will be created.

```
$ df
Mounted on    Filesystem         Avail/Total    Files      Status
/tmp          (OMVS.TMP)         1393432/1396800 4294967248 Available
/u/ibmuser    (OMVS.U.IBMUSER)   1248/1728      4294967281 Available
/usr/lpp/rdz  (OMVS.LPP.FEK)     3062/43200     4294967147 Available
/var          (OMVS.VAR)         27264/31680    4294967054 Available
```

Sample setup

The following sample setup shows the required configuration to support these requirements:

- 500 simultaneous client connections
- 300 simultaneous MVS builds (batch job)
- 200 simultaneous CARMA connections (using the CRASTART startup method)
- 3 hour inactivity time-out
- disallow usage of z/OS UNIX
- SCLM Developer Toolkit and File Manager Integration are not used
- Foresee an average Java heap usage of 5 MB
- Users have unique z/OS UNIX UIDs

Thread pool count

By default, Developer for system z tries to add 60 users to a single thread pool. However, our requirements indicate that the inactivity time-out will be active. Table 25 on page 71 shows that this will add 1 thread per connected client. This thread is a timer thread, and thus constantly active. This will prevent RSE from putting 60 users in a single thread pool, as 60*(16+1)=1020, and `maximum.threads` is set to 1000 by default.

We could increase `maximum.threads`, but due to the requirement to have on average 5 MB of Java heap per user, we choose to lower `maximum.clients` to 50. This keeps us within the default 256 MB maximum Java heap size (5*50 = 250).

With 50 clients per thread pool and the need to support 500 connections, we now know we will need 10 thread pool address spaces.

Determine minimum limits

Using the formulas shown earlier in this chapter and the criteria stated at the beginning of this section, we can determine the resource usage that must be accommodated.

- Address space count - maximum

  3 + A + N*(x + y + z) + (2 + N*0.01)

  3 + 10 + 500*1 + 200*1 + 300*1 + (2 + 500*0.01) = 1020
- Address space count - per user

  x + y + z

  1 + 1 + 1 = 3
- Process count - maximum

7 + 2*A + N*(x + y + z) + (10 + N*0.05)

7 + 2*10 + 500*2 + 200*1 + 300*0 + (10 + 500*0.05) = 1562

- Process count - per user

  (x + y + z) + 5*s

  (2 + 1 + 0) + 5*0 = 3
- Thread count - RSE thread pool

  9 + N*(16 + x + y + z) + (20 + N*0.1)

  9 + 60*(16 + 1 + 4 + 0) + (20 + 60*0.1) = 1295
- Thread count - JES Job Monitor

  3 + N

  3 + 500 = 503
- User IDs

  500 + 3 = 503

  The 3 extra user IDs are for STCJMON, STCLOCK and STCRSE, the Developer for System z started task user IDs.

Defining limits

Now that the resource usage numbers are known, we can customize the limiting directives with appropriate values.

- /etc/rdz/rsed.envvars
  - Xmx256m

    not changed
  - Dmaximum.clients=50
  - Dmaximum.threads=1000

    not changed
  - Dminimum.threadpool.process=10

    This change is optional; RSE will start new thread pools as needed
  - DHIDE_ZOS_UNIX=true
  - DDSTORE_IDLE_SHUTDOWN_TIMEOUT=10800000
- FEK.#CUST.PARMLIB(FEJJCNFG)
  - MAX_THREADS=503
- SYS1.PARMLIB(BPXPRMxx)
  - MAXPROCSYS(2500)

    1562 minimum, added extra buffer for tasks other than Developer for System z
  - MAXPROCUSER(25)

    not changed, minimum 3
  - MAXTHREADS(1500)

    must be minimum 503 (for JES Job Monitor) if THREADSMAX in the OMVS segment of user ID STCRSE is used to set the limit for RSE (minimum 1295)
  - MAXTHREADTASKS(1500)

must be minimum 503 (for JES Job Monitor) if THREADSMAX in the OMVS segment of user ID STCRSE is used to set the limit for RSE (minimum 1295)

  - MAXUIDS(700)

    503 minimum, added extra buffer for tasks other than Developer for System z

  - MAXASSIZE(209715200)

    not changed (200 MB system default), we use ASSIZEMAX in the OMVS segment of user ID STCRSE

- SYS1.PARMLIB(IEASYSxx)
  - MAXUSER=2000

    1020 minimum, added extra buffer for tasks other than Developer for System z

- OMVS segment of user ID STCRSE
  - ASSIZEMAX(2147483647)

    2 GB

Monitor resource usage

After activating the system limits as documented in "Defining limits" on page 91, we can start monitoring the resource usage by Developer for System z to see if adjustment of some variables is needed. Figure 21 shows the resource usage after 495 users logged on. (The example in the figure shows just the logging on. No user actions are indicated in the example.)

```
BPXM023I (STCRSE)
 ProcessId(16779764) Memory Usage(10%) Clients(50) Order(1)
 ProcessId(67108892) Memory Usage(16%) Clients(50) Order(2)
 ProcessId(67108908) Memory Usage(10%) Clients(50) Order(3)
 ProcessId(67108898) Memory Usage(16%) Clients(50) Order(4)
 ProcessId(67108916) Memory Usage(16%) Clients(50) Order(5)
 ProcessId(67108897) Memory Usage(16%) Clients(50) Order(6)
 ProcessId(67108921) Memory Usage(16%) Clients(50) Order(7)
 ProcessId(83886146) Memory Usage(16%) Clients(50) Order(8)
 ProcessId(67108920) Memory Usage(16%) Clients(50) Order(9)
 ProcessId(3622    ) Memory Usage(8%) Clients(45) Order(10)

 Jobname     Cpu time      Storage     EXCP
 --------   -----------    -------  ----------
 JMON              1.74     43.0M         2753
 LOCKD            10.05     31.9M        24621
 RSED              6.65     40.1M        41780
 RSED1             8.17    187.0M        76566
 RSED2            13.04    184.9M        78946
 RSED3            17.77    181.1M        76347
 RSED4            11.63    174.9M        74638
 RSED5            15.27    172.9M        72883
 RSED6            13.85    180.8M        75031
 RSED7             9.79    174.3M        76636
 RSED8            21.59    176.1M        70583
 RSED8            18.88    184.7M        76953
 RSED9             9.52    189.8M        80490
```

Figure 21. Resource usage of sample setup

Chapter 6. Performance considerations

z/OS is a highly customizable operating system, and (sometimes small) system changes can have a huge impact on the overall performance. This chapter highlights some of the changes that can be made to improve the performance of Developer for System z.

Refer to the *MVS Initialization and Tuning Guide* (SA22-7591) and *UNIX System Services Planning* (GA22-7800) for more information on system tuning.

Use zFS file systems

zFS (zSeries® File System) and HFS (Hierarchical File System) are both UNIX file systems that can be used in a z/OS UNIX environment. However, zFS provides the following features and benefits:

- Performance gains in many customer environments when accessing files approaching 8K in size that are frequently accessed and updated. The access performance of smaller files is equivalent to that of HFS.
- Read-only cloning of a file system in the same data set. The cloned file system can be made available to users to provide a read-only point-in-time copy of a file system. This is an optional feature that is available only in a non-sysplex environment.
- zFS is the strategic z/OS UNIX file system. The HFS functionality has been stabilized, and enhancements to the file system will be for zFS only.

Refer to *UNIX System Services Planning* (GA22-7800) to learn more about zFS.

Avoid use of STEPLIB

Each z/OS UNIX process that has a `STEPLIB` that is propagated from parent to child or across an exec will consume about 200 bytes of Extended Common Storage Area (ECSA). If no `STEPLIB` environment variable is defined, or when it is defined as `STEPLIB=CURRENT`, z/OS UNIX propagates all currently active `TASKLIB`, `STEPLIB`, and `JOBLIB` allocations during a fork(), spawn(), or exec() function.

Developer for System z has a default of `STEPLIB=NONE` coded in `rsed.envvars`, as described in `rsed.envvars`, configuration file. For the reasons mentioned previously, you should not change this directive and you should place the targeted data sets in LINKLIST or LPA (Link Pack Area) instead.

Improve access to system libraries

Certain system libraries and load modules are heavily used by z/OS UNIX and application development activities. Improving access to these, such as adding them to the Link Pack Area (LPA) can improve your system performance. Refer to *MVS Initialization and Tuning Reference* (SA22-7592) for more information on changing the `SYS1.PARMLIB` members described as follows:

Language Environment (LE) runtime libraries

When C programs (including the z/OS UNIX shell) are run, they frequently use routines from the Language Environment (LE) runtime library. On average, about 4

MB of the runtime library are loaded into memory for every address space running a LE-enabled program, and copied on every fork.

CEE.SCEELPA

The CEE.SCEELPA data set contains a subset of the LE runtime routines, which are heavily used by z/OS UNIX. You should add this data set to SYS1.PARMLIB(LPALSTxx) for maximum performance gain. By doing so, the modules are read from disk only once and are stored in a shared location.

Note: Add the following statement to SYS1.PARMLIB(PROGxx) if you prefer to add the load modules into dynamic LPA (Link Pack Area):

```
LPA ADD MASK(*) DSNAME(CEE.SCEELPA)
```

It is also advised to place the LE runtime libraries CEE.SCEERUN and CEE.SCEERUN2 in LINKLIST, by adding the data sets to SYS1.PARMLIB(LNKLSTxx) or SYS1.PARMLIB(PROGxx). This eliminates z/OS UNIX STEPLIB overhead and there is reduced input/output due to management by LLA and VLF, or similar products.

Note: Add the C/C++ DLL class library CBC.SCLBDLL also to LINKLIST for the same reasons.

If you decide not to put these libraries in LINKLIST, then you must set up the appropriate STEPLIB statement in rsed.envvars, as described in rsed.envvars, configuration file. Although this method always uses additional virtual storage, you can improve performance by defining the LE runtime libraries to LLA or a similar product. This reduces the I/O that is needed to load the modules.

Application development

On systems where application development is the primary activity, performance may also benefit if you put the linkage editor into dynamic LPA, by adding the following lines to SYS1.PARMLIB(PROGxx):

```
LPA ADD MODNAME(CEEBINIT,CEEBLIBM,CEEEV003,EDCZV) DSNAME(CEE.SCEERUN)
LPA ADD MODNAME(IEFIB600,IEFXB603) DSNAME(SYS1.LINKLIB)
```

For C/C++ development, you can also add the CBC.SCCNCMP compiler data set to SYS1.PARMLIB(LPALSTxx).

The preceding statements are samples of possible LPA candidates, but the needs at your site may vary. Refer to *Language Environment Customization* (SA22-7564) for information on putting other LE load modules into dynamic LPA. Refer to *UNIX System Services Planning* (GA22-7800) for more information on putting C/C++ compiler load modules into dynamic LPA.

Improving performance of security checking

To improve the performance of security checking done for z/OS UNIX, define the BPX.SAFFASTPATH profile in the FACILITY class of your security software. This reduces overhead when doing z/OS UNIX security checks for a wide variety of operations. These include file access checking, IPC access checking, and process ownership checking. Refer to *UNIX System Services Planning* (GA22-7800) for more information on this profile.

Note: Users do not need to be permitted to the BPX.SAFFASTPATH profile.

Workload management

Each site has specific needs, and can customize the z/OS operating system to get the most out of the available resources to meet those needs. With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, should be given to the work to meet its goal.

Developer for System z performance can be balanced by setting the correct goals for its processes. Some general guidelines are listed as follows:

- When used, assign the APPC transaction to a TSO performance group.
- Assign a started task performance group (SYSSTC) to the Developer for System z server address spaces: JES Job Monitor (JMON), Lock daemon (LOCKD), RSE daemon (RSED), and RSE thread pools (RSEDx).

Refer to *MVS Planning Workload Management* (SA22-7602) for more information about this subject.

Fixed Java heap size

With a fixed-size heap, no heap expansion or contraction occurs and this can lead to significant performance gains in some situations. However, using a fixed-size heap is usually not a good idea, because it delays the start of garbage collection until the heap is full, at which point it will be a major task. It also increases the risk of fragmentation, which requires a heap compaction. Therefore, use fixed-size heaps only after proper testing or under the direction of the IBM support center. Refer to *Java Diagnostics Guide* (SC34-6650) for more information on heap sizes and garbage collection.

The initial and maximum heap size of a z/OS Java Virtual Machine (JVM) can be set with the `-Xms` (initial) and `-Xmx` (maximum) Java command-line options.

In Developer for System z, Java command-line options are defined in the `_RSE_JAVAOPTS` directive of `rsed.envvars`, as described in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC23-7658).

```
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Xms128m -Xmx128m"
```

Java -Xquickstart option

Note: Java -Xquickstart is only useful if you use the REXEC/SSH alternate startup method for RSE server. This method is documented in "(Optional) Using REXEC (or SSH)" in the *Host Configuration Guide* (SC23-7658).

The -Xquickstart option can be used for improving startup time of some Java applications. -Xquickstart causes the JIT (Just In Time) compiler to run with a subset of optimizations; that is, a quick compile. This quick compile allows for improved startup time.

The -Xquickstart option is appropriate for shorter running applications, especially those where execution time is not concentrated into a small number of methods. -Xquickstart can degrade performance if it is used on longer-running applications that contain hot methods.

To enable the -Xquickstart option for the RSE server, add the following directive to the end of `rsed.envvars`:

```
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Xquickstart"
```

Class sharing between JVMs

The IBM Java Virtual Machine (JVM) version 5 and higher allows you to share bootstrap and application classes between JVMs by storing them in a cache in shared memory. Class sharing reduces the overall virtual memory consumption when more than one JVM shares a cache. Class sharing also reduces the startup time for a JVM after the cache has been created.

The shared class cache is independent of any active JVM and persists beyond the lifetime of the JVM that created the cache. Because the shared class cache persists beyond the lifetime of any JVM, the cache is updated dynamically to reflect any modifications that might have been made to JARs or classes on the file system.

The overhead to create and populate a new cache is minimal. The JVM startup cost in time for a single JVM is typically between 0 and 5% slower compared with a system not using class sharing, depending on how many classes are loaded. JVM startup time improvement with a populated cache is typically between 10% and 40% faster compared with a system not using class sharing, depending on the operating system and the number of classes loaded. Multiple JVMs running concurrently will show greater overall startup time benefits.

Refer to the *Java SDK and Runtime Environment User Guide* to learn more about class sharing.

Enable class sharing

To enable class sharing for the RSE server, add the following directive to the end of `rsed.envvars`. The first statement defines a cache named RSE with group access and it allows the RSE server to start even if class sharing fails. The second statement is optional and it sets the cache size to 6 megabytes (system default is 16 MB). The third statement adds the class sharing parameters to the Java startup options.

```
_RSE_CLASS_OPTS=-Xshareclasses:name=RSE,groupAccess,nonFatal
#_RSE_CLASS_OPTS="$_RSE_CLASS_OPTS -Xscmx6m
_RSE_JAVAOPTS="$_RSE_JAVAOPTS $_RSE_CLASS_OPTS"
```

Note: As mentioned in “Cache security,” all users using the shared class must have the same primary group ID (GID). This means that the users must have the same default group defined in the security software, or that the different default groups have the same GID in their OMVS segment.

Cache size limits

The maximum theoretical shared cache size is 2 GB. The size of cache you can specify is limited by the amount of physical memory and swap space available to the system. Because the virtual address space of a process is shared between the shared class cache and the Java heap, increasing the maximum size of the Java heap will reduce the size of the shared class cache you can create.

Cache security

Access to the shared class cache is limited by operating system permissions and Java security permissions.

By default, class caches are created with user-level security, so only the user that created the cache can access it. On z/OS UNIX, there is an option, groupAccess, which gives access to all users in the primary group of the user that created the cache. However, regardless of the access level used, a cache can only be destroyed by the user that created it or by a root user (UID 0).

Refer to *Java SDK and Runtime Environment User Guide* to learn more about extra security options using a Java SecurityManager.

SYS1.PARMLIB(BPXPRMxx)

Some of the SYS1.PARMLIB(BPXPRMxx) settings affect shared classes performance. Using the wrong settings can stop shared classes from working. These settings might also have performance implications. For further information about performance implications and use of these parameters, refer to *MVS Initialization and Tuning Reference* (SA22-7592) and *UNIX System Services Planning* (GA22-7800). The most significant BPXPRMxx parameters that affect the operation of shared classes are the following:

- MAXSHAREPAGES, IPCSHMSPAGES, IPCSHMMPAGES and IPCSHMNSEGS

  These settings affect the amount of shared memory pages available to the JVM. The shared page size for a 31-bit z/OS UNIX system service is fixed at 4 KB. Shared classes try to create a 16 MB cache by default. Therefore set IPCSHMMPAGES greater than 4096.

  If you set a cache size using -Xscmx, the JVM will round up the value to the nearest megabyte. You must take this into account when setting IPCSHMMPAGES on your system.
- IPCSEMNIDS and IPCSEMNSEMS

  These settings affect the amount of semaphores available to UNIX processes. Shared classes use IPC semaphores to communicate between the JVMs.

Disk space

The shared class cache requires disk space to store identification information about the caches that exist on the system. This information is stored in /tmp/javasharedresources. If the identification information directory is deleted, the JVM cannot identify the shared classes on the system and must recreate the cache.

Cache management utilities

The Java -Xshareclasses line command can take a number of options, some of which are cache management utilities. Three of them are shown in the following sample ($ is the z/OS UNIX prompt). Refer to *Java SDK and Runtime Environment User Guide* for a complete overview of supported command-line options.

```
$ java -Xshareclasses:listAllCaches
Shared Cache            OS shmid        in use          Last detach time
RSE                     401412          0               Mon Jun 18 17:23:16 2007

Could not create the Java virtual machine.

$ java -Xshareclasses:name=RSE,printStats

Current statistics for cache "RSE":


base address       = 0x0F300058
end address        = 0x0F8FFFF8
allocation pointer = 0x0F4D2E28

cache size         = 6291368
```

```
free bytes         = 4355696
ROMClass bytes     = 1912272
Metadata bytes     = 23400
Metadata % used    = 1%

# ROMClasses       = 475
# Classpaths       = 4
# URLs             = 0
# Tokens           = 0
# Stale classes    = 0
% Stale classes    = 0%

Cache is 30% full

Could not create the Java virtual machine.

$ java -Xshareclasses:name=RSE,destroy
JVMSHRC010I Shared Cache "RSE" is destroyed
Could not create the Java virtual machine.
```

Note:

- Cache utilities perform the required operation on the specified cache without starting the JVM, so the "`Could not create the Java virtual machine.`" message is normal.
- A cache can be destroyed only if all JVMs using it have shut down, and the user issuing the command has sufficient permissions.

Chapter 7. 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

Developer for System z clients version 8.0.1 and higher 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.

Since version 8.0.3, 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.

`pushtoclient.properties` tells the client if these functions are enabled, and where the related data is stored. See "(Optional) pushtoclient.properties, Host-based client control" in the *Host Configuration Guide* (SC23-7658) for more information.

Typically, z/OS systems, developer workstations, and development projects are managed by different groups of people. Push-to-client design follows this principle and assigns specific duties to each group:

- The z/OS system programmer controls the location of the push-to-client metadata, the basic security aspects, and whether push-to-client is active.
- The client administrator maintains the content of the push-to-client metadata by using the Developer for System z client to create one or more client

configurations, and by using IBM Installation Manager to create the response files used to update the Developer for System z client.

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

See the Developer for System z Information Center (`http://publib.boulder.ibm.com/infocenter/ratdevz/v8r0/index.jsp`) for details about how the client administrator and 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 `FEK.PTC.*` LDAP groups.
- A security administrator maintains access lists to `FEK.PTC.*` security profiles. A developer can be authorized to none, one, or more profiles.

Primary system

Push-to-client is designed to store system-specific data per system, while maintaining common (global) data on a single system (the primary system) to reduce management effort. The primary system is identified by the `primary.system` directive in `pushtoclient.properties`. The default is `false`.

Ensure you have one, and only one, system defined as the primary system. Developer for System z client administrators are not able to export global configuration data unless the target system is a primary system. Developer for System z clients might show erratic behavior when connecting to multiple primary systems with out-of-sync configurations.

The only-one rule does not apply when multiple systems share the Developer for System z configuration (`/etc/rdz`) and push-to-client metadata (`/var/rdz/pushtoclient`). Since the configuration is shared, all systems involved are identified as the primary system. But as long as all systems also share the metadata, this duplication is not an issue.

Push-to-client metadata

Metadata location

The `pushtoclient.folder` directive in `pushtoclient.properties` identifies the base directory where the push-to-client metadata is stored. The default is `/var/rdz/pushtoclient`.

The base directory holds the root push-to-client configuration file, `keymapping.xml`. All other metadata is in subdirectories.

Most subdirectories are created dynamically when the client administrator exports the push-to-client workspace configuration. These subdirectories group the metadata by subject, such as mappings and preferences. As more Developer for System z client components become eligible to be managed by push-to-client, more subdirectories are created dynamically. See the export wizard in the Developer for

System z client (**File > Export... > Rational Developer for System z > Configuration Files**) to learn what is stored in these subdirectories.

Some subdirectories are created during initial host customization. These subdirectories hold data that is maintained manually by the client administrator or development project manager.

- `/var/rdz/pushtoclient/projects/` holds the host-based project definition files. The actual location is specified in `/var/rdz/pushtoclient/keymapping.xml`, which is created and maintained by a Developer for System z client administrator. The files within are maintained by a project manager or lead developer.
- `/var/rdz/pushtoclient/install/` holds configuration files used to update the client product version upon connection to the host. The actual location is specified in `/var/rdz/pushtoclient/keymapping.xml`, which is created and maintained by a Developer for System z client administrator. The files within are maintained by a client administrator.
- `/var/rdz/pushtoclient/install/responsefiles/` holds configuration files used to update the client product version upon connection to the host. The actual location is specified in `/var/rdz/pushtoclient/keymapping.xml`, which is created and maintained by a Developer for System z client administrator. The files within are maintained by a client administrator.

See "Customization setup" in the "Basic customization" chapter of the *Host Configuration Guide* (SC23-7658) for more information about the creation of these subdirectories.

Metadata security

By default (see the `file.permission` directive in `pushtoclient.properties`), all files and directories created in the base directory receive permission bitmask 775 (`rwxrwxr-x`), which allows the owner and the owner's default group read and write access to the directory structure and the files within. Everyone else has only read access to the directory structure and the files within.

It is important that the correct owner UID (user ID) and GID (group ID) are set for these directories before starting with the push-to-client setup.

The following sample RACF commands create a new group (RDZADMIN), assign it a unique GID (2), and make it the default group for user ID RDZADM1, which also receives a unique UID (6).

```
ADDGROUP RDZADMIN OWNER(IBMUSER) SUPGROUP(SYS1) –
  DATA('RATIONAL DEVELOPER FOR SYSTEM Z - CLIENT ADMIN')
ALTGROUP RDZADMIN OMVS(GID(2))
CONNECT RDZADM1 GROUP(RDZADMIN) AUTH(USE)
ALTUSER RDZADM1 DFLTGRP(RDZADMIN) OMVS(UID(6))
```

The following sample **chown** z/OS UNIX command changes the owner and group of `/var/rdz/pushtoclient` and everything in it to `RDZADM1` and `RDZADMIN` respectively. The command should be executed by a super-user (UID 0) to avoid permission problems.

```
chown –R rdzadm1:rdzadmin /var/rdz/pushtoclient
```

The following sample **chmod** z/OS UNIX command changes the permission bitmask of `/var/rdz/pushtoclient` and everything in it to 775. Execute it to ensure that any manual addition to the directory follows the logic used by Developer for System z. The command should be executed by a super-user (UID 0) to avoid permission problems.

```
chmod –R 775 /var/rdz/pushtoclient
```

See *Security Server RACF Command Language Reference* (SA22-7687) for more information about the sample RACF commands. See *UNIX System Services Command Reference* (SA22-7802) for more information about the sample z/OS UNIX commands. See "z/OS UNIX directory structure" on page 11 for additional information.

Metadata space usage

The push-to-client metadata uses a reasonably small amount of disk space in z/OS UNIX, because the bulk of the metadata is UTF-8 encoded XML files. Note that the product code used for the client update scenarios can be stored anywhere on the network; it does not have to be stored in z/OS UNIX, because the related push-to-client metadata (called response files) point the client to the correct location.

Client configuration control

When a Developer for System z client (version 8.0.1 and higher) connects to the host, it reads the definitions in `pushtoclient.properties`. If directive `config.enabled` is enabled, the client compares its current configuration to the definitions in the push-to-client metadata. If differences are found, the client starts a wizard that pulls the required data and activates the setup as dictated by push-to-client.

The `reject.config.updates` directive in `pushtoclient.properties` controls whether a user is allowed to reject the configuration updates push-to-client is about to deliver.

A Developer for System z client (version 8.0.1 and higher) has a wizard, to be used by the client administrator, that can export the current configuration, which in turn is imported by all Developer for System z clients through push-to-client. Note that this function is available in all clients, so you should ensure that only client administrators have write permission to the z/OS UNIX directories that hold the push-to-client metadata (`/var/rdz/pushtoclient`).

Version 8.0.3 or higher is required for both client and host to enable group support, as documented in "Multiple developer groups" on page 103.

Client version control

When a Developer for System z client (version 8.0.1 and higher) connects to the host, it reads the definitions in `pushtoclient.properties`. If directive `product.enabled` is enabled, the client compares its current product version to the definitions in the push-to-client metadata. If differences are found, the client starts a wizard that pulls the required data and activates the setup as dictated by push-to-client.

The `reject.product.updates` directive in `pushtoclient.properties` controls whether a user is allowed to reject the product updates push-to-client is about to deliver.

Version 8.0.3 or higher is required for both client and host to enable group support, as documented in "Multiple developer groups" on page 103.

Multiple developer groups

Since version 8.0.3, 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.

Activation

Support for multiple developer groups, each with their own client configuration and client update requirements, is enabled by assigning the desired value to the related directives (`config.enabled` and `product.enabled`) in `pushtoclient.properties`, as shown in Table 29.

*Table 29. Push-to-client group support matrix for *.enabled*

*.enabled value	Function enabled	Multiple groups supported
False	No	No
True	Yes	No
LDAP	Yes	Yes, based on membership of LDAP groups `FEK.PTC.*.ENABLED.sysname.devgroup`
SAF	Yes	Yes, based on permit to security profiles `FEK.PTC.*.ENABLED.sysname.devgroup`

Note that when the function is enabled (this includes the `TRUE` value), developers are always part of a default group. A developer can be part of none, one, or multiple additional groups.

Rejecting the updates can also be made conditional, as shown in Table 30.

Table 30. Push-to-client group support matrix for reject..updates*

reject.*.updates value	Function enabled
False	No
True	Yes
LDAP	Depends on LDAP group membership `FEK.PTC.REJECT.*.UPDATES.sysname`
SAF	Depends on permit to security profile `FEK.PTC.REJECT.*.UPDATES.sysname`

Note that the directives in `pushtoclient.properties` work independently from each other. You can assign any supported value to any directive. There is no requirement to keep settings alike.

See “LDAP-based group selection” on page 106 and “SAF-based group selection” on page 111 for details about the required setup for the respective function. See “(Optional) pushtoclient.properties, Host-based client control” in the *Host Configuration Guide* (SC23-7658) for more information about enabling multiple group support.

Group concatenations

When the `*.enabled` function is enabled (this includes the `TRUE` value) in `pushtoclient.properties`, developers are always part of a default group for the related function. A developer can be part of none, one, or multiple additional groups.

To limit the complexity of applying changes defined in multiple groups, Developer for System z limits which definitions will be used, based on a selection made by the user.

Table 31. Push-to-client group concatenations

Additional groups	Definitions used
None	Default
One	Default or (default + group)
Multiple	Default or (default + 1 group)

Developer for System z uses the following logic when building and applying the change set:

1. Take the updates, if any, specified in the default definitions.
2. Take the updates specified in the selected group definition, if any, changing the default ones if they are already there.
3. Apply the updates on the client.

Note: Updates can consist of delete, add, and overlay actions.

Workspace binding

Even though a developer can be part of multiple groups simultaneously, the developer's active workspace cannot. The active workspace must be bound to a specific group (which can be the default group) to receive configuration or product updates. Once the bind is done, it cannot be undone. A new workspace must be created if a new group binding is required.

When a workspace that has no group binding connects to the host, and `config.enabled` (or `product.enabled`) indicates the push-to-client function is active, Developer for System z queries all groups to determine to which groups the user belongs to and prompts the user to select a group for the related function. Upon successive connections, only the selected group is queried to see if the group membership is still valid.

The `reject.*.updates` directives do not work with multiple groups, so their setup is simpler and does not require workspace binding. When an update is present, Developer for System z determines if the user is allowed to reject the update, and acts accordingly.

Group metadata location

As documented in "Metadata location" on page 100, all push-to-client metadata is stored in a directory structure on top of `/var/rdz/pushtoclient/` when using a setup without group support. The same data layout is maintained when group support is activated, but with a slightly different interpretation of the base directory, `/var/rdz/pushtoclient/`:

- Existing data in `/var/rdz/pushtoclient/` is interpreted as the data for the default group. Exporting to the default group creates or updates the metadata in

/var/rdz/pushtoclient/. This interpretation ensures compatibility with version 8.0.1 and version 8.0.2 clients, which are push-to-client enabled but do not support multiple groups.

- Exporting to a group creates or updates the metadata in /var/rdz/pushtoclient/grouping/<devgroup>/, as if this were the base directory instead of /var/rdz/pushtoclient/. The <devgroup> value matches the group name assigned to a specific group of developers.

Initial product customization creates the grouping/ directory in /var/rdz/pushtoclient/. The client administrator is responsible for adding the <devgroup>/ directories to /var/rdz/pushtoclient/grouping/.

Note that during initial product customization, the projects/, install/, and install/responsefiles/ directories are created in /var/rdz/pushtoclient/. The client administrator must repeat these make-directory actions in /var/rdz/pushtoclient/grouping/<devgroup>/ if there is a need for group-specific product upgrade scenarios or group-specific host-based projects.

The following sample z/OS UNIX command sequence creates the subdirectories with the correct permission bitmask. The commands should be executed by the client administrator to avoid ownership problems.

```
saved_umask=$(umask)
umask 0000
cd /var/rdz/pushtoclient/grouping/
mkdir –m775 <devgroup>
cd <devgroup>
mkdir –m775 install
mkdir –m775 install/responsefiles
mkdir –m775 projects
umask $saved_umask
```

See *UNIX System Services Command Reference* (SA22-7802) for more information about the sample z/OS UNIX commands.

Setup steps

Setting up support for multiple developer groups requires some coordination between the z/OS system programmer, the client administrator, and the administrator managing the selection criteria (LDAP or security administrator). In the following description of the work flow, the security administrator manages the selection criteria:

1. The client administrator asks the security administrator for input about existing grouping setup for developers. Reusing the existing setup speeds up and simplifies push-to-client setup.
2. The client administrator determines how he wants to structure the multiple group support, and who should be part of these push-to-client groups.

   Note:

   - There is always a default configuration set and a default product update scenario.
   - Push-to-client change sets can include delete, add, and overlay actions.
   - Push-to-client change sets can be empty.
   - A developer can be part of none, one, or multiple push-to-client groups.
   - The client administrator must be a member of each push-to-client group.

3. The client administrator and the security administrator agree on the push-to-client group names to be used.
4. The client administrator creates the

   `/var/rdz/pushtoclient/grouping/<devgroup>`

   directory for each push-to-client group.

   Note: The permission bits for this directory should be `775` (`drwxrwxr-x`).
5. The security administrator does the required initial setup to define the push-to-client selection criteria profiles and adds the push-to-client groups to the access lists.

   Note:
   - The selection criteria structures must be defined with at least the client administrator on the access list before the client administrator can create the related push-to-client metadata.
   - For initial setup, only the client administrator should be on the access list for a push-to-client group. This to avoid Developer for system z clients receiving setups that are under construction.
6. The z/OS system programmer activates multiple group support by adjusting `pushtoclient.properties`.

   Note: The `*.enabled` directives must be enabled before the client administrator can create the related push-to-client metadata.
7. The client administrator creates the workspaces for each group and exports them to the host using the respective group names. The client administrator also creates the required response files to create group-specific product update scenarios.
8. The security administrator adds the developers to the push-to-client groups, activating push-to-client for the developers.

LDAP-based group selection

Although LDAP (Lightweight Directory Access Protocol) is the name of a TCP/IP based protocol, it is commonly used to describe a set of distributed directory services. Like a database, a directory is a structured collection of records. Developer for System z can use an LDAP server as a simple hierarchical database, where groups hold one or more members.

When using definitions in your LDAP server as selection mechanism (the `LDAP` value is specified for directives in `pushtoclient.properties`), Developer for System z verifies membership of the group names listed in Table 32 to determine which developer groups the user belongs to, and whether a user is allowed to reject updates.

Table 32. Push-to-client LDAP information

Group name (cn=)	Result
FEK.PTC.CONFIG.ENABLED.sysname.devgroup	Client accepts configuration updates for the specified group
FEK.PTC.PRODUCT.ENABLED.sysname.devgroup	Client accepts product updates for the specified group
FEK.PTC.REJECT.CONFIG.UPDATES.sysname	User can reject configuration updates

Table 32. Push-to-client LDAP information (continued)

Group name (cn=)	Result
FEK.PTC.REJECT.PRODUCT.UPDATES.sysname	User can reject product updates

The `devgroup` value matches the group name assigned to a specific group of developers. Note that the group name is visible on Developer for System z clients.

The `sysname` value matches the system name of the target system.

LDAP schema

The LDAP schema must satisfy the following rules:

1. Each push-to-client group must be defined as group in the schema.
2. Each user must be defined as user in the schema.
3. A group entry has the references to the user entries that belong to its own group.

Figure 22 is a sample LDAP definition for a group and user, expressed in LDIF format.

Note: LDAP Data Interchange Format (LDIF) is a standard text format for representing LDAP objects and LDAP updates. Files containing LDIF records are used to transfer data between directory servers or as input by LDAP utilities.

```
# Group Definition
dn: cn=FEK.PTC.CONFIG.ENABLED.CDFMVS08.GROUPA,o=PTC,c=DeveloperForZ
objectClass: groupOfUniqueNames
objectClass: top
cn: FEK.PTC.CONFIG.ENABLED.CDFMVS08.GROUPA
description: Project A
uniqueMember: uid=mborn,ou=Users,dc=example,dc=com

# User Definition
dn: uid=mborn,ou=Users,dc=example,dc=com
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: uidObject
objectClass: top
cn: May Born
sn: Born
uid: mborn
facsimiletelephonenumber: +1 800 982 6883
givenname: May
mail: mborn@example.com
ou: Users
```

Figure 22. Sample LDAP schema definition

LDAP server selection

There is a wide selection of commercial and free LDAP servers available. One example is the IBM Tivoli® Directory Server (`http://www-01.ibm.com/software/tivoli/products/directory-server/`). There is also a wide selection of command-line and GUI-based tools to manage an LDAP server.

As mentioned in "LDAP schema," each user must be defined to the LDAP server. To reduce management effort, it is best to place the push-to-client schema on an

LDAP server that already has access to all user definitions. For example, you can use IBM Tivoli Directory Server active on z/OS using an SDBM database (which is a wrapper for your security database).

Depending on site policies, the push-to-client schema on the LDAP server might be managed by the client administrator. This arrangement reduces collaboration needs, and possible delays and communication errors.

An argument in favor of LDAP management by the client administrator is that the push-to-client schema does not hold anything confidential or security-related. When user definitions are available to the LDAP server through other schemas, the Developer for System z LDAP objects just determine which choices a developer has in selecting a workspace layout and automatic Developer for System z client product upgrades.

LDAP server location

Any database server that supports the LDAP protocol can be used to host the Developer for System z push-to-client schema. Therefore, Developer for System z allows you to specify the information needed to connect to the LDAP server. You can also specify the suffix that makes the database unique within the LDAP server.

rsed.envvars directive	Default
_RSE_LDAP_SERVER	Local host system
_RSE_LDAP_PORT	389
_RSE_LDAP_PTC_GROUP_SUFFIX	"O=PTC,C=DeveloperForZ"

Note that TCP/IP security measures, like firewalls, might stop the (host-based) RSE server from contacting the LDAP server. Contact your TCP/IP administrator with the following information to ensure the LDAP server can be reached:

- LDAP server TCP/IP address or DNS name
- LDAP server port number
- LDAP uses the TCP protocol
- LDAP server is contacted by the host-based RSE server
- RSE server is active in an `RSEDx` address space, where `RSED` is the RSE started task name and `x` is a random one-digit number

Sample setup

Assume a company has Developer for System z active on system CDFMVS08. IBM Tivoli Directory Server, also active on CDFMVS08, is used as LDAP server. The LDAP server is configured as described in “Adding push-to-client back-end to LDAP” on page 109.

The following users use Developer for System z:

- Developers who work on banking applications, user ID BNK010 -> BNK014
- Developers who work on insurance applications, user ID INS010 -> INS014
- A Developer for System z client administrator, user ID RDZADM1

Each group of developers requires specific client configuration files, and all developers are subject to the same client version control. Unlike client administrators, developers are not allowed to reject any of the changes push-to-client presents.

The client administrator and LDAP administrator agreed on using group names BANKING and INSURANCE for the configuration updates.

Adding push-to-client back-end to LDAP

In this example, updates are made to IBM Tivoli Directory Server on z/OS, currently using only an SDBM database (security database wrapper) by adding an LDBM database (z/OS UNIX files) to host the push-to-client schema.

1. Add the LDBM back-end section to the LDAP configuration file.

```
# filename ds.conf
# restart GLDSRV started task to pick up changes

# global section
adminDN "cn=LDAP admin"
adminPW password
listen ldap://:389
schemaPath /etc/ldap

# SDBM back-end section (RACF)
database SDBM GLDBSD31/GLDBSD64
suffix "cn=RACF,o=IBM,c=US"

# LDBM back-end section (z/OS UNIX files)
database LDBM GLDBLD31/GLDBLD64 LDBM-RDZ
suffix "o=PTC,c=DeveloperForZ"
databaseDirectory /var/ldap/ldbm/rdz
```

2. Stop and start LDAP started task, `GRDSRV`, to pick up the configuration changes.
3. Create the `/var/ldap/ldbm/rdz` directory.

```
mkdir -p /var/ldap/ldbm/rdz
```

4. Update LDAP schema to add the LDBM back-end.

```
ldapmodify -D "cn=LDAP admin" -w password -f
/usr/lpp/ldap/etc/schema.user.ldif

ldapmodify -D "cn=LDAP admin" -w password -f
/usr/lpp/ldap/etc/schema.IBM.ldif
```

5. Add the root entry to the LDBM back-end.

```
ldapadd -D "cn=LDAP admin" -w password -f
/u/ibmuser/ptc_root.ldif
```

   where `/u/ibmuser/ptc_root.ldif` holds the following:

```
dn: o=PTC,c=DeveloperForZ
objectclass: top
objectclass: organization
o: PTC
```

Initial LDAP group setup

Add the different LDAP group objects to the schema, and make the client administrator part of each of them. The user definition for the RDZADM1 user ID is pulled from the RACF schema.

```
ldapadd -D "cn=LDAP admin" -w password -f /u/ibmuser/ptc_setup.ldif
```

where `/u/ibmuser/ptc_setup.ldif` holds the following:

```
#  banking workspace configuration
dn: cn=FEK.PTC.CONFIG.ENABLED.CDFMVS08.BANKING,o=PTC,c=DeveloperForZ
objectclass: groupOfUniqueNames
cn: FEK.PTC.CONFIG.ENABLED.CDFMVS08.BANKING
description: Developer for System z push-to-client
# give client administrator access
uniqueMember: racfID=RDZADM1,profileType=user,cn=RACF,o=IBM,c=US
```

```
#  insurance workspace configuration
dn: cn=FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE,o=PTC,c=DeveloperForZ
objectclass: groupOfUniqueNames
cn: FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE
description: Developer for System z push-to-client
# give client administrator access
uniqueMember: racfID=RDZADM1,profileType=user,cn=RACF,o=IBM,c=US

#  reject configuration updates
dn: cn=FEK.PTC.REJECT.CONFIG.UPDATES.CDFMVS08,o=PTC,c=DeveloperForZ
objectclass: groupOfUniqueNames
cn: FEK.PTC.REJECT.CONFIG.UPDATES.CDFMVS08
description: Developer for System z push-to-client
# give client administrator access
uniqueMember: racfID=RDZADM1,profileType=user,cn=RACF,o=IBM,c=US

#  reject product updates
dn: cn=FEK.PTC.REJECT.PRODUCT.UPDATES.CDFMVS08,o=PTC,c=DeveloperForZ
objectclass: groupOfUniqueNames
cn: FEK.PTC.REJECT.PRODUCT.UPDATES.CDFMVS08
description: Developer for System z push-to-client
# give client administrator access
uniqueMember: racfID=RDZADM1,profileType=user,cn=RACF,o=IBM,c=US
```

Add developers to LDAP groups

Add the developers to the LDAP group objects. The user definitions for the user IDs are pulled from the RACF schema.

```
ldapmodify -D "cn=LDAP admin" -w password -f /u/ibmuser/ptc_add.ldif
```

where `/u/ibmuser/ptc_add.ldif` holds the following:

```
# banking workspace configuration
dn: cn=FEK.PTC.CONFIG.ENABLED.CDFMVS08.BANKING,o=PTC,c=DeveloperForZ
changeType: modify
add: uniqueMember
uniqueMember: racfID=BNK010,profileType=user,cn=RACF,o=IBM,c=US
uniqueMember: racfID=BNK011,profileType=user,cn=RACF,o=IBM,c=US
uniqueMember: racfID=BNK012,profileType=user,cn=RACF,o=IBM,c=US
uniqueMember: racfID=BNK013,profileType=user,cn=RACF,o=IBM,c=US
uniqueMember: racfID=BNK014,profileType=user,cn=RACF,o=IBM,c=US

# insurance workspace configuration
dn: cn=FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE,o=PTC,c=DeveloperForZ
changeType: modify
add: uniqueMember
uniqueMember: racfID=INS010,profileType=user,cn=RACF,o=IBM,c=US
uniqueMember: racfID=INS011,profileType=user,cn=RACF,o=IBM,c=US
uniqueMember: racfID=INS012,profileType=user,cn=RACF,o=IBM,c=US
uniqueMember: racfID=INS013,profileType=user,cn=RACF,o=IBM,c=US
uniqueMember: racfID=INS014,profileType=user,cn=RACF,o=IBM,c=US
```

pushtoclient.properties

```
# BANKING and INSURANCE have different configuration needs
config.enabled=LDAP
# everyone receives product updates
product.enabled=TRUE
# only RDZADMIN can reject configuration updates
reject.config.updates=LDAP
# only RDZADMIN can reject product updates
reject.product.updates=LDAP
```

rsed.envvars

No updates are required because the defaults are used:

- _RSE_LDAP_SERVER=CDFMVS08.RALEIGH.IBM.COM

- _RSE_LDAP_PORT=389
- _RSE_LDAP_PTC_GROUP_SUFFIX="o=PTC,c=DeveloperForZ"

/var/rdz/pushtoclient/*install

While exporting the workspace configuration for groups BANKING and INSURANCE, the export wizard creates the `/var/rdz/pushtoclient/grouping/<devgroup>/` directories, and the directory structure behind it.

- /var/rdz/pushtoclient/grouping/BANKING/*
- /var/rdz/pushtoclient/grouping/INSURANCE/*

Because there are no individualized product upgrade scenarios, the client administrator does not need to create or update the `install/` and `install/responsefiles/` subdirectories in `/var/rdz/pushtoclient/grouping/<devgroup>`.

The client administrator must create the response files needed for product updates in the default-group directory, `/var/rdz/pushtoclient/install/responsefiles/`.

SAF-based group selection

SAF (Security Access Facility) is an interface to access any z/OS security product. Developer for System z can use this interface to query your security product and retrieve push-to-client related information.

When using definitions in your security database as selection mechanism (the SAF value is specified for directives in `pushtoclient.properties`), Developer for System z verifies access permits to the profiles listed in Table 33 to determine which developer groups the user belongs to, and whether a user is allowed to reject updates.

Table 33. Push-to-client SAF information

FACILITY profile	Fixed length	Required access	Result
FEK.PTC.CONFIG.ENABLED. sysname.devgroup	23	READ	Client accepts configuration updates for the specified group
FEK.PTC.PRODUCT.ENABLED. sysname.devgroup	24	READ	Client accepts product updates for the specified group
FEK.PTC.REJECT.CONFIG. UPDATES.sysname	30	READ	User can reject configuration updates
FEK.PTC.REJECT.PRODUCT. UPDATES.sysname	31	READ	User can reject product updates

Note: Developer for System z assumes a user has no access authorization when your security software indicates it cannot determine whether or not a user has access authorization to a profile. An example of this is when the profile is not defined.

The `devgroup` value matches the group name assigned to a specific group of developers. Note that the group name is visible on Developer for System z clients.

The sysname value matches the system name of the target system.

The "Fixed length" column documents the length of the fixed part of the related security profile.

By default, Developer for System z expects the FEK.PTC.* profiles to be in the FACILITY security class. Note that profiles in the FACILITY class are limited to 39 characters. If the sum of the length of the fixed profile part (FEK.PTC.<key>.) and the length of the site-specific profile part (sysname or sysname.devgroup) exceeds this number, you can place the profiles in another class and instruct Developer for System z to use this class instead. To do that, uncomment _RSE_FEK_SAF_CLASS in rsed.envvars and provide the desired class name.

Sample setup

Assume a company has Developer for System z active on system CDFMVS08. The RACF security database is shared among multiple systems and the following groups are defined in the security database:

- DEVBANK : developers who work on banking applications
- DEVINSUR : developers who work on insurance applications
- RDZADMIN : Developer for System z client administrators

Each group of developers requires specific client configuration files, and all developers are subject to the same client version control. Unlike client administrators, developers are not allowed to reject any of the changes push-to-client presents. The reject rule is valid for all systems, in preparation for future expansion.

The client and security administrator agree to use push-to-client group names BANKING and INSURANCE for the configuration updates.

Security definition

```
# allow RDZADMIN and DEVBANK to select push-to-client group BANKING
RDEFINE FACILITY (FEK.PTC.CONFIG.ENABLED.CDFMVS08.BANKING) –
  UACC(NONE) DATA('RATIONAL DEVELOPER FOR SYSTEN Z - PUSH-TO-CLIENT')
PERMIT FEK.PTC.CONFIG.ENABLED.CDFMVS08.BANKING CLASS(FACILITY) –
  ID(RDZADMIN DEVBANK) ACCESS(READ)

# allow RDZADMIN and DEVINSUR to select push-to-client group INSURANCE
RDEFINE FACILITY (FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE) –
  UACC(NONE) DATA('RATIONAL DEVELOPER FOR SYSTEN Z - PUSH-TO-CLIENT')
PERMIT FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE CLASS(FACILITY) –
  ID(RDZADMIN DEVINSUR) ACCESS(READ)

# RDZADMIN can reject configuration updates on any system
RDEFINE FACILITY (FEK.PTC.REJECT.CONFIG.UPDATES.*) –
  UACC(NONE) DATA('RATIONAL DEVELOPER FOR SYSTEN Z - PUSH-TO-CLIENT')
PERMIT FEK.PTC.REJECT.CONFIG.UPDATES.* CLASS(FACILITY) –
  ID(RDZADMIN) ACCESS(READ)

# RDZADMIN can reject product updates on any system
RDEFINE FACILITY (FEK.PTC.REJECT.PRODUCT.UPDATES.*) –
  UACC(NONE) DATA('RATIONAL DEVELOPER FOR SYSTEN Z - PUSH-TO-CLIENT')
PERMIT FEK.PTC.REJECT.CONFIG.UPDATES.* CLASS(FACILITY) –
  ID(RDZADMIN) ACCESS(READ)

# activate changes
SETROPTS RACLIST(FACILITY) REFRESH
```

pushtoclient.properties

```
# BANKING and INSURANCE have different configuration needs
config.enabled=SAF
# everyone receives product updates
product.enabled=TRUE
# only RDZADMIN can reject configuration updates
reject.config.updates=SAF
# only RDZADMIN can reject product updates
reject.product.updates=SAF
```

rsed.envvars

No updates are required because the defaults are used:

```
_RSE_FEK_SAF_CLASS=FACILITY
```

/var/rdz/pushtoclient/*install

While exporting the workspace configuration for groups BANKING and INSURANCE, the export wizard creates the `/var/rdz/pushtoclient/grouping/<devgroup>/` directories, and the directory structure behind it.

- var/rdz/pushtoclient/grouping/BANKING/*
- /var/rdz/pushtoclient/grouping/INSURANCE/*

Because there are no individualized product upgrade scenarios, the client administrator does not need to create or update the `install/` and `install/responsefiles/` subdirectories in `/var/rdz/pushtoclient/grouping/<devgroup>/`.

The client administrator must create the response files needed for product updates in the default-group directory, `/var/rdz/pushtoclient/install/responsefiles/`.

Grace period for rejecting changes

Assume that while the sample setup is active, a Developer for System z fix-pack with important fixes becomes available, but the timing of a banking project is such that various developers might be very weary of changing anything on their workstation right now.

To resolve the issue, the security administrator can grant all DEVBANK developers a grace period in which they can choose to postpone (reject) the update.

Setting up the grace period is a very simple process:

```
# start of grace period
PERMIT FEK.PTC.REJECT.PRODUCT.UPDATES.* CLASS(FACILITY) -
  ID(DEVBANK) ACCESS(READ)

# activate changes
SETROPTS RACLIST(FACILITY) REFRESH
```

At the end of the grace period, the additional authority can be removed again:

```
# end of grace period
PERMIT FEK.PTC.REJECT.PRODUCT.UPDATES.* CLASS(FACILITY) -
  ID(DEVBANK) DELETE

# activate changes
SETROPTS RACLIST(FACILITY) REFRESH
```

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/rdz/pushtoclient/keymapping.xml`, and is `/var/rdz/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/rdz/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 discussed in “Multiple developer groups” on page 103. This eligibility means that host-based projects can be defined also in `/var/rdz/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 8. CICSTS considerations

Traditionally, the role of defining resources to CICS has been the domain of the CICS administrator. There has been a reluctance to allow the application developer to define CICS resources for various reasons:

- Most CICS resource definitions have many parameters that because of their complexity, interrelationship with other resource definitions, and shop standards require CICS administrator knowledge to define correctly. Incorrect definitions can cause unexpected results that might impact the entire CICS region.
- Most customer shops provide CICS development and test environments that must be available for shared use by multiple application groups and developers. Many customer shops have Service Level Agreements in place for these environments. Meeting these agreements requires strict control of the environments.

Developer for System z addresses these issues by allowing CICS administrators to control CICS resource definition defaults, and to control the display properties of a CICS resource definition parameter by means of the CICS Resource Definition (CRD) server, which is part of Application Deployment Manager.

For example, the CICS administrator can supply certain CICS resource definition parameters that might not be updated by the application developer. Other CICS resource definition parameters may be updatable, with or without supplied defaults, or the CICS resource definition parameter can be hidden to avoid unnecessary complexity.

Once the application developer is satisfied with the CICS resource definitions they may be installed immediately in the running CICS test environment, or the definitions may be exported in a manifest for further editing and approval by a CICS administrator. The CICS administrator can use the administrative utility (batch utility) or the Manifest Processing tool to implement resource definition changes.

Note: The Manifest Processing tool is a plugin for IBM CICS Explorer.

Refer to "(Optional) Application Deployment Manager" in the *Host Configuration Guide* (SC23-7658) for more information on the tasks needed to set up Application Deployment Manager on your host system.

Customizing Application Deployment Manager adds the following services to Developer for System z:

- (on the client) IBM CICS Explorer® provides an Eclipse-based infrastructure to view and manage CICS resources and enables greater integration between CICS tools
- (on the client) CICS Resource Definition (CRD) editor
- (on the host) CICS Resource Definition (CRD) server, which runs as a CICS application

The Application Deployment Manager CICS Resource Definition (CRD) server consists of the CRD server itself, a CRD repository, associated CICS resource definitions, and, when using the Web Service interface, Web Service bind files, and

a sample pipeline message handler. The CRD server must run in a Web Owning Region (WOR), which is referenced in the Developer for System z documentation as the CICS primary connection region.

Refer to the Developer for System z Information Center (`http://publib.boulder.ibm.com/infocenter/ratdevz/v8r0/index.jsp`) to learn more about the services Application Deployment Manager available in the current release of Developer for System z.

RESTful versus Web Service

CICS Transaction Server provides in version 4.1 and higher support for an HTTP interface designed using Representational State Transfer (RESTful) principles. This RESTful interface is now the strategic CICSTS interface for use by client applications. The older Web Service interface has been stabilized, and enhancements will be for the RESTful interface only.

Application Deployment Manager follows this statement of direction and requires the RESTful CRD server for all services that are new to Developer for System version 7.6 or higher.

The RESTful and Web Service interfaces can be active concurrently in a single CICS region, if desired. In this case, there will be two CRD servers active in the region. Both servers will share the same CRD repository. Note that CICS will issue some warnings about duplicate definitions when the second interface is defined to the region.

Primary versus non-primary connection regions

A CICS test environment may consist of several Multi-Region Option (MRO) connected regions. Over time, unofficial designations have been used to categorize these regions. Typical designations are Terminal Owning Region (TOR), Web Owning Region (WOR), Application Owning Region (AOR), and Data Owning Region (DOR).

A Web Owning Region is used to implement CICS Web Services support, and the Application Deployment Manager CICS Resource Definition (CRD) server must run in this region. This region is known to Application Deployment Manager as the CICS primary connection region. The CRD client implements a Web service connection to the CICS primary connection region.

CICS non-primary connection regions are all other regions that the CRD server can service. This service includes viewing resources using IBM CICS Explorer and defining resources using the CICS resource definition editor.

If CICSPlex® SM Business Application Services (BAS) is used to manage the CICS resource definitions of the CICS primary connection region, then all other CICS regions managed by BAS can be serviced by the CRD server.

CICS regions not managed by BAS require additional changes to be serviceable by the CRD server.

CICS resource install logging

Actions done by the CRD server against the CICS resources are logged in the CICS CSDL TD queue, which typically points to DD MSGUSR of your CICS region.

If CICSPlex SM Business Application Services (BAS) is used to manage your CICS resource definitions, then the CICSPlex SM EYUPARM directive BASLOGMSG must be set to (YES) for the logging to be created.

Application Deployment Manager security

CRD repository security

The CRD server repository VSAM data set holds all the default resource definitions and must therefore be protected against updates, but developers must be allowed to read the values stored here. Refer to “Define data set profiles” on page 35 for sample RACF commands to protect the CRD repository.

Pipeline security

When a SOAP message is received by CICS through the Web Service interface, the message is processed by a pipeline. A pipeline is a set of message handlers that are executed in sequence. CICS reads the pipeline configuration file to determine which message handlers should be invoked in the pipeline. A message handler is a program in which you can perform special processing of Web service requests and responses.

Application Deployment Manager provides a sample pipeline configuration file that specifies the invocation of a message handler and a SOAP header processing program.

The pipeline message handler (ADNTMSGH) is used for security by processing the user ID and password in the SOAP header. ADNTMSGH is referenced by the sample pipeline configuration file and must therefore be placed into the CICS RPL concatenation.

Transaction security

CPIH is the default transaction ID under which an application invoked by a pipeline will run. Typically, CPIH is set for a minimal level of authorization.

Developer for System z supplies multiple transactions that are used by the CRD server when defining and inquiring CICS resources. These transaction IDs are set by the CRD server, depending on the requested operation. Refer to "(Optional) Application Deployment Manager" in the *Host Configuration Guide* (SC23-7658) for more information on customizing the transaction IDs.

Transaction	Description
ADMS	For requests from the Manifest Processing tool to change CICS resources. Typically, this is intended for CICS administrators. This transaction requires a high level or authorization.
ADMI	For requests that define, install or uninstall CICS resources. This transaction might require a medium level of authorization, depending on your site policies.
ADMR	For all other requests that retrieve CICS environmental or resource information. This transaction might require a minimal level of authorization, depending on your site policies.

Some, or all, of the resource definition requests done by the CRD server transactions should be secured. At a minimum, the update commands (update default Web service parameters, default descriptor parameters, and file name to data set name binding) should be secured to prevent all but CICS administrators from issuing these commands used to set global resource defaults.

When the transaction is attached, CICS resource security checking, if enabled, insures that the user ID is authorized to run the transaction ID.

Resource checking is controlled by the `RESSEC` option in the transaction that is running, the `RESSEC` system initialization parameter, and for the CRD server, the `XPCT` system initialization parameter.

Resource checking occurs only if the `XPCT` system initialization parameter has a value other than `NO` and either the `RESSEC` option in the `TRANSACTION` definition is `YES` or the `RESSEC` system initialization parameter is `ALWAYS`.

The following RACF commands give a sample on how the CRD server transactions can be protected. Refer to *RACF Security Guide for CICSTS* for more information on defining CICS security.

- ```
  RALTER GCICSTRN SYSADM UACC(NONE) ADDMEM(ADMS)
  ```
- ```
  PERMIT SYSADM CLASS(GCICSTRN) ID(#cicsadmin)
  ```
- ```
  RALTER GCICSTRN DEVELOPER UACC(NONE) ADDMEM(ADMI)
  ```
- ```
  PERMIT DEVELOPER CLASS(GCICSTRN) ID(#cicsdeveloper)
  ```
- ```
  RALTER GCICSTRN ALLUSER UACC(READ) ADDMEM(ADMR)
  ```
- ```
  SETROPTS RACLIST(TCICSTRN) REFRESH
  ```

SSL encrypted communication

SSL encryption of the data stream is supported when the Application Deployment Manager client uses the Web Services interface to invoke the CRD server. The usage of SSL for this communication is controlled by the `SSL(YES)` keyword in the CICSTS `TCPIPSERVICE` definition, as documented in *RACF Security Guide for CICSTS*.

Resource security

CICSTS provides the ability to protect resources and the commands to manipulate them. Certain Application Deployment Manager actions might fail if security is active, but not configured completely (for example, granting permissions to manipulate new resource types).

Upon function failure in Application Deployment Manager, examine the CICS log for messages like the following, and take corrective action, as documented in *RACF Security Guide for CICSTS*.

```
DFHXS1111 %date %time %applid %tranid Security violation by user
%userid at netname %portname for resource %resource in class
%classname. SAF codes are (X'safresp',X'safreas'). ESM codes are
(X'esmresp',X'esmreas').
```

Administrative utility

Developer for System z provides the administrative utility to let CICS administrators provide the default values for CICS resource definitions. These defaults can be read-only, or can be editable by the application developer.

The administrative utility provides the following functions:

- CICSPlex name for CICSPlex managed test environments
- CICSPlex SM staging group name
- Manifest export rule setting
- CICS resource attribute defaults and display permissions
- CICS logical to physical binding used for VSAM data set definitions

The administrative utility is invoked by sample job `ADNJSPAU` in data set `FEK.#CUST.JCL`. The usage of this utility requires UPDATE access to the CRD repository.

`ADNJSPAU` is located in `FEK.#CUST.JCL`, unless the z/OS system programmer specified a different location when he customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See "Customization setup" in the *Host Configuration Guide* (SC23-7658) for more details.

Note: The CRD repository must be closed in CICS before running the ADNJSPAU job. The repository can be opened again after job completion. For example, after signing on to CICS, enter the following commands to close and open the file, respectively:

- `CEMT S FILE(ADNREPF0) CLOSED`
- `CEMT S FILE(ADNREPF0) OPEN`

Input control statements are used to update the CRD repository for a CICS test environment, for which the following general syntax rules apply:

- An asterisk in position 1 indicates a comment line.
- A DEFINE command must begin in position 1, followed by a single space, followed by a valid keyword, such as `TRANSACTION`.
- A keyword value must immediately follow a keyword. No intervening spaces are permitted. The only exception is for display permission keywords `UPDATE`, `PROTECT`, and `HIDDEN`, which have no values.
- Keyword values are enclosed within parenthesis.
- A keyword and its value must be contained on a single line.

The following sample definitions follow the structure of the DFHCSDUP commands, as defined in the *CICS Resource Definition Guide for CICSTS*. The only difference is the insertion of the following display permission keywords used to group the attribute values into three permission sets:

`UPDATE`	Attributes following this keyword will be updatable by an application developer using Developer for System z. This is also the default for omitted attributes.

PROTECT	Attributes following this keyword will display, but be protected from update by an application developer using Developer for System z.
HIDDEN	Attributes following this keyword will not display, and will be protected from update by an application developer using Developer for System z.

See the following ADNJSPAU code sample.

```
//ADNJSPAU JOB <JOB PARAMETERS>
//*
//ADNSPAU  EXEC PGM=ADNSPAU,REGION=1M
//STEPLIB  DD DISP=SHR,DSN=FEK.SFEKLOAD
//ADMREP   DD DISP=OLD,DSN=FEK.#CUST.ADNREPF0
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
*
* CICSPlex SM parameters
*
DEFINE CPSMNAME(        )
*DEFINE STAGINGGROUPNAME(ADMSTAGE)
*
* Manifest export rule
*
DEFINE MANIFESTEXPORTRULE(installOnly)
*
* CICS resource definition defaults
* Omitted attributes default to UPDATE.
*
*   DB2TRAN default attributes
*
DEFINE DB2TRAN()
       UPDATE DESCRIPTION()
              ENTRY()
              TRANSID()
*
*   DOCTEMPLATE default attributes
*
DEFINE DOCTEMPLATE()
       UPDATE DESCRIPTION()
              TEMPLATENAME()
              FILE() TSQUEUE() TDQUEUE() PROGRAM() EXITPGM()
              DDNAME(DFHHTML) MEMBERNAME()
              HFSFILE()
              APPENDCRLF(YES) TYPE(EBCDIC)
*
*   File default attributes
*
DEFINE FILE()
       UPDATE  DESCRIPTION()
               RECORDSIZE() KEYLENGTH()
               RECORDFORMAT(V) ADD(NO)
               BROWSE(NO) DELETE(NO) READ(YES) UPDATE(NO)
               REMOTESYSTEM() REMOTENAME()
       PROTECT DSNAME() RLSACCESS(NO) LSRPOOLID(1) STRINGS(1)
               STATUS(ENABLED) OPENTIME(FIRSTREF)
               DISPOSITION(SHARE) DATABUFFERS(2) INDEXBUFFERS(1)
               TABLE(NO) MAXNUMRECS(NOLIMIT)
               READINTEG(UNCOMMITTED) DSNSHARING(ALLREQS)
               UPDATEMODEL(LOCKING) LOAD(NO)
               JNLREAD(NONE) JOURNAL(NO)
               JNLSYNCREAD(NO) JNLUPDATE(NO)
               JNLADD(NONE) JNLSYNCWRITE(YES)
               RECOVERY(NONE) FWDRECOVLOG(NO)
               BACKUPTYPE(STATIC)
               PASSWORD() NSRGROUP()
               CFDTPOOL() TABLENAME()
```

Figure 23. ADNJSPAU - CICSTS administrative utility

```
*
* Mapset default attributes
*
DEFINE MAPSET()
       UPDATE  DESCRIPTION()
       PROTECT RESIDENT(NO) STATUS(ENABLED)
               USAGE(NORMAL) USELPACOPY(NO)
** Processtype default attributes
*
DEFINE PROCESSTYPE()
       UPDATE  DESCRIPTION()
               FILE(BTS)
       PROTECT STATUS(ENABLED)
               AUDITLOG() AUDITLEVEL(OFF)
*
* Program default attributes
*
DEFINE PROGRAM()
       UPDATE  DESCRIPTION()
               CEDF(YES) LANGUAGE(LE370)
               REMOTESYSTEM() REMOTENAME() TRANSID()
       PROTECT API(CICSAPI) CONCURRENCY(QUASIRENT)
               DATALOCATION(ANY) DYNAMIC(NO)
               EXECKEY(USER) EXECUTIONSET(FULLAPI)
               RELOAD(NO) RESIDENT(NO)
               STATUS(ENABLED) USAGE(NORMAL) USELPACOPY(NO)
       HIDDEN JVM(NO) JVMCLASS() JVMPROFILE(DFHJVMPR)
*
* TDQueue default attributes
*
DEFINE TDQUEUE()
       UPDATE  DESCRIPTION()
               TYPE(INTRA)
* Extra partition parameters
               DDNAME() DSNAME()
               REMOTENAME() REMOTESYSTEM() REMOTELENGTH(1)
               RECORDSIZE() BLOCKSIZE(0) RECORDFORMAT(UNDEFINED)
               BLOCKFORMAT() PRINTCONTROL() DISPOSITION(SHR)
* Intra partition parameters
               FACILITYID() TRANSID() TRIGERRLEVEL(1)
               USERID()
* Indirect parameters
               INDIRECTNAME()
       PROTECT WAIT(YES) WAITACTION(REJECT)
* Extra partition parameters
               DATABUFFERS(1)
               SYSOUTCLASS() ERROROPTION(IGNORE)
               OPENTIME(INITIAL) REWIND(LEAVE) TYPEFILE(INPUT)
* Intra partition parameters
               ATIFACILITY(TERMINAL) RECOVSTATUS(NO)
```

Figure 24. ADNJSPAU - CICSTS administrative utility (Part 2 of 3)

```
*
* Transaction default attributes
*
DEFINE TRANSACTION()
       UPDATE  DESCRIPTION()
               PROGRAM()
               TWASIZE(0)
               REMOTESYSTEM() REMOTENAME() LOCALQ(NO)
       PROTECT PARTITIONSET() PROFILE(DFHCICST)
               DYNAMIC(NO) ROUTABLE(NO)
               ISOLATE(YES) STATUS(ENABLED)
               RUNAWAY(SYSTEM) STORAGECLEAR(NO)
               SHUTDOWN(DISABLED)
               TASKDATAKEY(USER) TASKDATALOC(ANY)
               BREXIT() PRIORITY(1) TRANCLASS(DFHTCL00)
               DTIMOUT(NO) RESTART(NO) SPURGE(NO) TPURGE(NO)
               DUMP(YES) TRACE(YES) CONFDATA(NO)
               OTSTIMEOUT(NO) WAIT(YES) WAITTIME(00,00,00)
               ACTION(BACKOUT) INDOUBT(BACKOUT)
               RESSEC(NO) CMDSEC(NO)
               TRPROF()
               ALIAS() TASKREQ()
               XTRANID() TPNAME() XTPNAME()
*
* URDIMAP attributes
*
DEFINE URIMAP()
       UPDATE  USAGE(CLIENT)
               DESCRIPTION()
               PATH(/required/path)
               TCPIPSERVICE()
               TRANSACTION()
               PROGRAM()
       PROTECT ANALYZER(NOANALYZER)
               ATOMSERVICE()
               CERTIFICATE()
               CHARACTERSET()
               CIPHERS()
               CONVERTER()
               HFSFILE()
               HOST(host.mycompany.com)
               HOSTCODEPAGE()
               LOCATION()
               MEDIATYPE()
               PIPELINE()
               PORT(NO)
               REDIRECTTYPE(NONE)
               SCHEME(HTTP)
               STATUS(ENABLED)
               TEMPLATENAME()
               USERID()
               WEBSERVICE()
*
* Optional file name to VSAM data set name binding
*
*DEFINE DSBINDING() DSNAME()
/*
```

Figure 25. ADNJSPAU - CICSTS administrative utility (Part 3 of 3)

Administrative utility migration notes

Developer for System z version 7.6.1 added URIMAP support to the Administrative utility. To be able to use the URIMAP support, the CRD repository

VSAM data set must be allocated with a maximum record size of 3000. Up till Developer for System z version 7.6.1, the sample CRD repository allocation job uses a maximum record size of 2000.

Follow these steps to enable the URIMAP support if you're using an older CRD repository:

1. Create a backup of your existing CRD repository, `FEK.#CUST.ADNREPF0`.
2. Delete the existing CRD repository.
3. Customize and submit job `FEK.SFEKSAMP(ADNVCRD)` to allocate and initialize a new CRD repository. Refer to the documentation within the member for customization instructions.
4. Customize and submit job `FEK.SFEKSAMP(ADNJSPAU)` to use the Administrative utility to populate the new CRD repository.

Note:

- Migrating the existing CRD repository is not necessary, because the Administrative utility replaces the complete contents of the CRD repository each time it is executed.
- There are no version compatibility issues with the CRD repository. All supported Developer for System z client and host code will work with either maximum record size. But URIMAP support will be disabled if the maximum record size is not 3000.

Administrative utility messages

The following messages are issued by the Administrative utility to the SYSPRINT DD. Messages `CRAZ1803E`, `CRAZ1891E`, `CRAZ1892E`, and `CRAZ1893E` contain file status, VSAM return, VSAM function, and VSAM feedback codes. VSAM return, function, and feedback codes are documented in *DFSMS Macro Instructions for Data Sets* (SC26-7408). File status codes are documented in *Enterprise COBOL for z/OS Language Reference* (SC27-1408).

CRAZ1800I
completed successfully on line <last control statement line number>

Explanation: The system programmer administrative utility completed successfully.

User response: None.

CRAZ1801W
completed with warnings on line <last control statement line number>

Explanation: The system programmer administrative utility completed with one or more warnings found when processing control statements.

User response: Check other warning messages.

CRAZ1802E
encountered an error on line < line number>

Explanation: The system programmer administrative utility encountered a severe error.

User response: Check other warning messages.

CRAZ1803E
Repository open error, status=<file status code> RC=<VSAM return code> FC=<VSAM function code> FB=<VSAM feedback code>

Explanation: The system programmer administrative utility encountered a severe error opening the CRD repository.

User response: Check VSAM status, return, function, and feedback codes.

CRAZ1804E
Unrecognized input record on line <line number>

Explanation: The system programmer administrative utility encountered an unrecognized input control statement.

User response: Check a **DEFINE** command was followed by a single space, followed by the keyword `CPSMNAME`, `STAGINGGROUPNAME`, `MANIFESTEXPORTRULE`, `DSBINDING`, `DB2TRAN`, `DOCTEMPLATE`, `FILE`, `MAPSET`, `PROCESSTYPE`, `PROGRAM`, `TDQUEUE`, or `TRANSACTION`.

CRAZ1805E
Processing keyword <keyword> on line <line number>

Explanation: The system programmer administrative utility is processing the DEFINE keyword input control statement.

User response: None.

CRAZ1806E
Invalid manifest export rule on line <line number>

Explanation: The system programmer administrative utility encountered an invalid manifest export rule.

User response: Check that the `MANIFESTEXPORTRULE` keyword value is "installOnly", "exportOnly", or "both".

CRAZ1807E
Missing DSNAME keyword on line <line number>

Explanation: The system programmer administrative utility was processing a `DEFINE DSBINDING` control statement which is missing the `DSNAME` keyword.

User response: Check that the `DEFINE DSBINDING` control statement contains the `DSNAME` keyword.

CRAZ1808E
Invalid keyword value for keyword <keyword> on line <line number>

Explanation: The system programmer administrative utility was processing a `DEFINE` control statement and encountered an invalid value for the named keyword.

User response: Check that the length and value of the named keyword is correct.

CRAZ1890W
Keyword syntax error on line <line number>

Explanation: The system programmer administrative utility was processing a `DEFINE` control statement and encountered a syntax error for a keyword or keyword value.

User response: Check that the keyword value is enclosed in parenthesis and immediately follows the keyword. The keyword and keyword value must both be contained on the same line.

CRAZ1891W

Repository duplicate key write error, status=<file status code> RC=<VSAM return code> FC=<VSAM function code> FB=<VSAM feedback code>

Explanation: The system programmer administrative utility encountered a duplicate key error writing to the CRD repository.

User response: Check VSAM status, return, function, and feedback codes.

CRAZ1892W

Repository write error, status=<file status code> RC=<VSAM return code> FC=<VSAM function code> FB=<VSAM feedback code>

Explanation: The system programmer administrative utility encountered a severe error writing to the CRD repository.

User response: Check VSAM status, return, function, and feedback codes.

CRAZ1893W

Repository read error, status=<file status code> RC=<VSAM return code> FC=<VSAM function code> FB=<VSAM feedback code>

Explanation: The system programmer administrative utility encountered a severe error reading from the CRD repository.

User response: Check VSAM status, return, function, and feedback codes.

Chapter 9. Customizing the TSO environment

This appendix is provided to assist you with mimicking a TSO logon procedure by adding DD statements and data sets to the TSO environment in Developer for System z.

The TSO Commands service

The TSO Commands service is the Developer for System z component which executes TSO and (batch) ISPF commands, and returns the result to the requesting client. These commands can be requested implicitly by the product, or explicitly by the user.

The sample members provided with Developer for System z create a minimal TSO/ISPF environment. If the developers in your shop need access to custom or third-party libraries, the z/OS system programmer must add the necessary DD statements and libraries to the TSO Commands service environment. Although the implementation is different in Developer for System z, the logic behind it is identical to the TSO logon procedure.

Note: The TSO Commands service is a non-interactive command-line tool, so commands or procedures that prompt for data or display ISPF panels will not work. A 3270 emulator, such as the Host Connect Emulator which is part of the Developer for System z client, is needed to execute these.

Access methods

Since version 7.1, Developer for System z provides a choice on how to access the TSO Commands service.

- ISPF's TSO/ISPF Client Gateway service, which requires a minimum ISPF service level. This is the default method used in the provided samples.
- An APPC transaction (as in pre-version 7.1 releases). This method is deprecated.

Note:

- ISPF's TSO/ISPF Client Gateway service replaces the SCLM Developer Toolkit function used in version 7.1.
- APPC usage by Developer for System z is marked deprecated. The APPC related information has been removed from this publication. For more information, refer to white paper *Using APPC to provide TSO command services* (SC14-7291), available in the Developer for System z library, `http://www.ibm.com/software/rational/products/developer/systemz/library/index.html`.

Check `rsed.envvars` to determine which access method is used for version 7.1 and higher hosts. If defaults were used during the configuration process, `rsed.envvars` resides in `/etc/rdz/`.

- If the `_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DTSO_SERVER=APPC"` statement is not present (or commented out), ISPF's TSO/ISPF Client Gateway service is used.
- If the `_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DTSO_SERVER=APPC"` statement is present (and not commented out), APPC is used.

Using the TSO/ISPF Client Gateway access method

ISPF.conf

The `ISPF.conf` configuration file (by default located in `/etc/rdz/`) defines the TSO/ISPF environment used by Developer for System z. There is only one active `ISPF.conf` configuration file, which is used by all Developer for System z users.

The main section of the configuration file defines the DD names and the related data set concatenations, like that in the following sample:

```
sysproc=ISP.SISPCLIB,FEK.SFEKPROC
ispmlib=ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
isplllib=ISP.SISPLOAD
myDD=HLQ1.LLQ1,HLQ2.LLQ2
```

- Each DD definition uses exactly one line (multi-line is not supported), and there are no line length limits.
- The definitions are not case sensitive, and any white space will be ignored.
- Comment lines start with an asterisk (*).
- DD names are followed by an equal sign (=), which in turn is followed by the data set concatenation. Multiple data set names are separated by a comma (,).
- Data set concatenations are searched in the order they are listed.
- Data sets must be fully qualified, without being enclosed in quotes (‘), and without the use of variables.
- All data sets are allocated with `DISP=SHR`.
- New DD names can be added at will, but must obey the (JCL) rules for DD names and may not conflict with other configuration parameters in `ISPF.conf`. Also, `ISPPROF` is allocated dynamically (`DISP=NEW,DELETE`) by the TSO/ISPF Client Gateway service.

Use existing ISPF profiles

By default, the TSO/ISPF Client Gateway creates a temporary ISPF profile for the TSO Commands service. However, you can instruct the TSO/ISPF Client Gateway z to use a copy of an existing ISPF profile. The key here is the `_RSE_CMDSERV_OPTS` statement in `rsed.envvars`.

```
#_RSE_CMDSERV_OPTS="$_RSE_CMDSERV_OPTS &ISPPROF=&SYSUID..ISPPROF"
```

Uncomment the statement (remove the leading pound sign (#) character) and provide the fully qualified data set name of the existing ISPF profile to use this facility.

The following variables can be used in the data set name:

- `&SYSUID.` to substitute the developer's user ID
- `&SYSPREF.` to substitute the developer's TSO prefix

Note:

- If the data set name passed in `"ISPPROF"` is invalid, a temporary, empty ISPF profile is used instead.
- The ISPF profile (both temporary and copied) is deleted at the end of the session. Changes made to the profile are not merged into the existing ISPF profile.

Using an allocation exec

The allocjob statement in ISPF.conf (which is commented out by default) points to an exec which can be used to provide further data set allocations by user ID.

```
*allocjob = ISP.SISPSAMP(ISPZISP2)
```

Uncomment the statement (remove the leading asterisk (*) character) and provide the fully qualified reference to the allocation exec to use this facility.

- The exec is executed after allocation of ISPPROF and the DDs defined in ISPF.conf, but before ISPF is initialized. Ensure that your allocation exec does not undo these definitions.
- 1 parameter is passed to the exec; the user ID of the caller.
- A sample exec CRAISPRX is provided in sample library FEK.#CUST.CNTL, unless you specified a different location when you customized and submitted job FEK.SFEKSAMP(FEKSETUP). See "Customization setup" in the *Host Configuration Guide* (SC23-7658) for more details.

Note: As the exec is called before ISPF is initialized, you cannot use **VPUT** and **VGET**. You can however create your own implementation of these functions using a PDS(E) or VSAM file.

Use multiple allocation execs

Although ISPF.conf only supports calling one allocation exec, there are no limits on that exec calling another exec. And the user ID of the client being passed as parameter opens the door to calling personalized allocation execs. You can, for example, check if member USERID'.EXEC(ALLOC)' exists and execute it.

An elaborate variation to this theme enables you to use the existing TSO logon procedures, as follows:

- Read a user-specific configuration file, such as USERID'.FEKPROF'.
- See which logon procedure is mentioned in the file.
- Read the mentioned procedure from SYS1.PROCLIB and parse it to find the DD statements and data set allocations within.
- Allocate the data set in a similar fashion as the real logon procedure.

Multiple ISPF.conf files with multiple Developer for System z setups

If the allocation exec scenarios described in the previous sections cannot handle your specific needs, you can create different instances of Developer for System z's RSE communication server, with each instance using its own ISPF.conf file. The main drawback of the method described below is that Developer for System z users must connect to different servers on the same host to get the desired TSO environment.

Note: Creating a second instance of the RSE server only requires duplicating and updating configuration files, startup JCL and started task definitions. A new installation of the product is not necessary, nor is any code duplicated.

```
$ cd /etc/rdz
$ mkdir /etc/rdz/tso2
$ cp rsed.envvars /etc/rdz/tso2
$ cp ISPF.conf /etc/rdz/tso2
$ ls /etc/rdz/tso2
ISPF.conf          rsed.envvars
$ oedit /etc/rdz/tso2/rsed.envvars
```

```
      -> change: _RSE_RSED_PORT=4037
      -> change: _CMDSERV_CONF_HOME=/etc/rdz/tso2
      -> change: -Ddaemon.log=/var/rdz/logs/tso2
      -> change: -Duser.log=/var/rdz/logs/tso2
      -> add at the END:
         # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
         CFG_BASE=/etc/rdz
         CLASSPATH=.:$CFG_BASE:$CLASSPATH
         # --
$ oedit /etc/rdz/tso2/ISPF.conf
      -> change: change as needed
```

The commands in the previous example copy the Developer for System z configuration files that require changes to a newly created `tso2` directory. The `_CMDSERV_CONF_HOME` variable in `rsed.envvars` must be updated to define the new `ISPF.conf` home directory , and `daemon.log` and `user.log` must be updated to define a new log location (which is created automatically if it does not exist). The `_RSE_RSED_PORT` update ensures that both the existing and the new RSE daemon will use unique port numbers. The `CLASSPATH` update ensures that RSE can find the configuration files that were not copied to `tso2`. The `ISPF.conf` file itself can be updated to match your needs. Note that the ISPF workarea (variable `_CMDSERV_WORK_HOME` in `rsed.envvars`) can be shared among both instances.

What is left now is creating a new started task for RSE that uses a new port number and the new `/etc/rdz/tso2` configuration files. Note that if `_RSE_RSED_PORT` is not changed in `rsed.envvars`, the new started task must specify a new port as startup argument.

Refer to the *IBM Rational Developer for System z Host Configuration Guide* (SC23-7658) for more information on the actions shown previously in this section.

Chapter 10. Running multiple instances

There are times that you want multiple instances of Developer for System z active on the same system, for example, when testing an upgrade. However, some resources such as TCP/IP ports cannot be shared, so the defaults are not always applicable. Use the information in this appendix to plan the coexistence of the different instances of Developer for System z, after which you can use this configuration guide to customize them.

Although it is possible to share certain parts of Developer for System z between two (or more) instances, it is advised NOT to do so, unless their software levels are identical and the only changes are in configuration members. Developer for System z leaves enough customization room to make multiple instances that do not overlap and we strongly advise you to use these features.

Note:

- `FEK` and `/usr/lpp/rdz` are the high-level qualifier and path used during the installation of the product. `FEK.#CUST`, `/etc/rdz` and `/var/rdz` are the default locations used during the customization of the product (see "Customization setup" in the *Host Configuration Guide* (SC23-7658) for more information)..
- You should install Developer for System z in a private file system (HFS or zFS) to ease deploying the z/OS UNIX parts of the product.
- If you can not use a private file system, you should use an archiving tool such as the z/OS UNIX `tar` command to transport the z/OS UNIX directories from system to system. This to preserve the attributes (such as program control) for the Developer for System z files and directories.

  Refer to *UNIX System Services Command Reference* (SA22-7802) for more information on the following sample commands to archive and restore the Developer for System z installation directory.

  – Archive: `cd /SYS1/usr/lpp/rdz; tar -cSf /u/userid/rdz.tar`
  – Restore: `cd /SYS2/usr/lpp/rdz; tar -xSf /u/userid/rdz.tar`

Identical setup across a sysplex

Developer for System z configuration files (and code) can be shared across different systems in a sysplex, with each system running its own identical copy of Developer for System z, if a few guidelines are obeyed. Note that this information is for stand-alone Developer for System z instances. Additional rules for the TCP/IP setup apply when using Distributed Dynamic VIPA to group multiple servers (each on a separate system) into one virtual server, as documented in “Distributed Dynamic VIPA” on page 51.

- The log files should end up in unique locations to avoid one system overwriting information from another. By routing the z/OS UNIX logs to specific locations with the `daemon.log` and `user.log` directives in `rsed.envvars`, you can share the configuration files if you mount a system specific z/OS UNIX file system on the specified path. This way, all logs are written to the same logical place, but due to the unshared file system underneath, they end up in different physical locations.
- Configuration-type directories like `/etc/rdz/` and `/var/rdz/pushtoclient/` can be shared across the sysplex, as Developer for System z uses them in read-only mode.

- Temporary data directories like /tmp/ and /var/rdz/WORKAREA/ must be unique per system, because temporary file names are not sysplex-aware.
- If you share the code, you should also share the configuration files to ensure you do not have some systems that are out of synchronization after applying maintenance.
- If you share an active /etc/rdz/pushtoclient.properties configuration file, you must also share the related metadata directory, /var/rdz/pushtoclient/.

Identical software level, different configuration files

In a limited set of circumstances, you can share all but (some of) the customizable parts. An example is providing non-SSL access for on-site usage, and SSL encoded communication for off-site usage.

Attention: The shared setup CANNOT be used safely to test maintenance, a technical preview, or a new release.

To set up another instance of an active Developer for System z installation, redo the customization steps for the parts that are different, using different data sets, directories, and ports to avoid overlapping the current setup.

In the SSL sample mentioned previously, the current RSE daemon setup can be cloned, after which the cloned setup can be updated. Next the RSE daemon startup JCL can be cloned and customized with a new TCP/IP port and the location of the updated configuration files. The MVS customizations (JES Job Monitor, and so on) can be shared between the SSL and non-SSL instances. This would result in the following actions:

```
$ cd /etc/rdz
$ mkdir /etc/rdz/ssl
$ cp rsed.envvars /etc/rdz/ssl
$ cp ssl.properties /etc/rdz/ssl
$ ls /etc/rdz/ssl/
rsed.envvars    ssl.properties
$ oedit /etc/rdz/ssl/rsed.envvars
      -> change: _RSE_RSED_PORT=4047
      -> change: -Ddaemon.log=/var/rdz/logs/ssl
      -> change: -Duser.log=/var/rdz/logs/ssl
      -> add at the END:
         # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
         CFG_BASE=/etc/rdz
         CLASSPATH=.:$CFG_BASE:$CLASSPATH
         # --
$ oedit /etc/rdz/ssl/ssl.properties
      -> change: change as needed
```

The commands in the preceding example copy the Developer for System z configuration files that require changes to a newly created ssl directory. The daemon.log and user.log variables in rsed.envvars must be updated to define a new log location (which is created automatically if it does not exist). The CLASSPATH update ensures that RSE can find the configuration files that were not copied to ssl. The ssl.properties file itself can be updated to match your needs.

What is left now is creating a new started task for RSE that uses a new port number and the new /etc/rdz/ssl configuration files.

Refer to the related sections in the *IBM Rational Developer for System z Host Configuration Guide* (SC23-7658) for more information on the actions shown previously in this section.

All other situations

When code changes are involved (maintenance, technical previews, new release), or your changes are fairly complex, you should do another installation of Developer for System z. This section describes possible points of conflict between the different installations.

The following list is a brief overview of items that must or are strongly advised to be different between the instances of Developer for System z:

- SMP/E CSI
- Installation libraries
- JES Job Monitor TCP/IP port, and thus its configuration file FEJJCNFG
- JES Job Monitor startup JCL
- APPC transaction name
- RSE configuration files, rsed.envvars, *.properties and *.conf
- RSE TCP/IP port
- RSE startup JCL

A more detailed overview is listed as follows:

- SMP/E CSI
  1. Install each instance of Developer for System z in a separate CSI. SMP/E will prevent a second install of the same FMID in a CSI, but will accept installing another FMID. If the second FMID is a newer version, it will delete the existing version of the product. If the second FMID is an older version, the install will fail due to duplicate part names.
- Installation libraries
  1. Install each instance of Developer for System z in separate data sets and directories. Keep in mind that you can only change the z/OS UNIX path by prefixing the IBM supplied default of `/usr/lpp/rdz`. A valid sample would be `/service/usr/lpp/rdz`.
  2. Customization setup job `FEK.SFEKSAMP(FEKSETUP)` creates the data sets and directories used to store configuration files. Since the configuration files must be unique, and to avoid overwriting existing customizations, you must use unique data set and directory names when submitting this job.
- Mandatory parts
  1. JES Job Monitor configuration file `FEK.#CUST.PARMLIB(FEJJCNFG)` holds the TCP/IP port number of JES Job Monitor and thus cannot be shared. The member itself can be renamed (if the JCL is updated also), so you can place all customized versions of this member in the same data set if you are not doing the updates in the install data set.
  2. JES Job Monitor startup JCL `FEK.#CUST.PROCLIB(JMON)` refers to `FEJJCNFG` and therefore cannot be shared either. After renaming the member (and the JOB card if you start it as a user job) you can place all JCL's in the same data set.
  3. The RSE configuration file `/etc/rdz/rsed.envvars` holds references to the install path, and optionally to the server log location, which requires it to be unique. The file name is mandatory, so you cannot keep the different copies in the same directory.

4. The `ISPF.conf` configuration file has a reference to `FEK.SFEKPROC(FEKFRSRV)`, the TSO Commands server. This is software level specific, so you must create an `ISPF.conf` file per instance.
5. All other z/OS UNIX based configuration files (such as `*.properties`) must reside in the same directory as `rsed.envvars` and thus cannot be shared, since `rsed.envvars` must be in an unshared location.
6. The RSE startup JCL `FEK.#CUST.PROCLIB(RSED)` cannot be shared since it defines the TCP/IP port number and it has a reference to the install and configuration directories, which must be unique. After renaming the member (and the JOB card if you start it as a user job) you can place all JCL's in the same data set.
7. The lock daemon startup JCL `FEK.#CUST.PROCLIB(LOCKD)` cannot be shared since it has a reference to the installation and configuration directories, which must be unique. After renaming the member (and the JOB card if you start it as a user job) you can place all JCLs in the same data set.

- Optional parts
  1. The REXEC and SSH TCP/IP ports can be shared without any restrictions.
  2. The APPC transaction has a reference to `FEK.SFEKPROC(FEKFRSRV)`, the TSO Commands server. This is software level specific, so you must create an APPC transaction per instance. Keep in mind that, since the APPC transaction name changes, the `_FEKFSCMD_TP_NAME_` variable must be defined in `rsed.envvars`.
  3. Some ELAXF* procedures have a reference to `hlq.SFEKLOAD`, the Developer for System z load library. See the note on JCLLIB in "ELAXF* remote build procedures" in the *Host Configuration Guide* (SC23-7658) for a possible solution on making different sets available to the users.
  4. To activate two instances of the DB2 stored procedure, the following tasks must be completed. Note however that this is a non-supported, as-is description:
     a. Copy `hlq.SFEKPROC(ELAXMREX)` to a differently named member, for example, `ELAXMRXX`.
     b. Copy sample member `hlq.SFEKSAMP(ELAXMSAM)` to a differently named member, for example, `ELAXMWDZ`.
     c. Change sample member `hlq.SFEKSAMP(ELAXMJCL)` to reflect these name changes, for example:

```
//SYSIN DD *
 CREATE PROCEDURE SYSPROC.ELAXMRXX
  ( IN  FUNCTION_REQUEST   VARCHAR(20)      CCSID EBCDIC
...
  , OUT RETURN_VALUE       VARCHAR(255)     CCSID EBCDIC )
 PARAMETER STYLE GENERAL  RESULT SETS 1
 LANGUAGE REXX            EXTERNAL NAME   ELAXMRXX
 COLLID   DSNREXCS        WLM ENVIRONMENT ELAXMWDZ
 PROGRAM TYPE MAIN        MODIFIES SQL DATA
 STAY RESIDENT NO         COMMIT ON RETURN NO
 ASUTIME NO LIMIT         SECURITY USER;

 COMMENT ON PROCEDURE SYSPROC.ELAXMRXX IS
 'PLI & COBOL PROCEDURE PROCESSOR (ELAXMRXX), INTERFACE LEVEL 0.01';

 GRANT EXECUTE ON PROCEDURE SYSPROC.ELAXMRXX TO PUBLIC;
//
```

     d. Proceed with the customization as described in "(Optional) DB2 stored procedure" in the *Host Configuration Guide* (SC23-7658), but with the new members.

e. The new WLM environment name (for example, ELAXMWDZ) must be used in the DB2 stored procedure wizard on the client.

5. Bidi support in CICS regions relies on a load library member and thus cannot be shared across releases. However, if the load module name is identical for all instances, you can share the most recent version between the instances, even across releases. Backward compatibility is not available if the load module's name has changed.
6. The Application Deployment Manager load modules that are included in CICS regions are backwards compatible, and thus the most recent version can be shared across releases.
7. The Application Deployment Manager CRD VSAM is backwards compatible, and thus the most recent version can be shared across releases.
8. The Application Deployment Manager CICS resource definitions are backwards compatible, and thus the most recent version can be shared across releases.
9. CARMA VSAMs could change between software levels, thus it is not advised to share these.

Chapter 11. Troubleshooting configuration problems

This chapter is provided to assist you with some common problems that you may encounter during your configuration of Developer for System z, and has the following sections:

The *Developer for System z Messages and Codes* (SC14-7497) publication documents messages and return codes generated by Developer for System z components.

More information is available through the Support section of the Developer for System z Web site (`http://www.ibm.com/software/rational/products/developer/systemz/`) where you can find Technotes that bring you the latest information from our support team.

In the Library section of the Web site (`http://www.ibm.com/software/rational/products/developer/systemz/library/index.html`) you can also find the latest version of the Developer for System z documentation, including whitepapers.

The Developer for System z Information Center (`http://publib.boulder.ibm.com/infocenter/ratdevz/v8r0/index.jsp`) documents the Developer for System z client, and how it interacts with the host (from a client's perspective).

Valuable information can also be found in the z/OS internet library, available at `http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Please notify us if you think that Developer for System z misses a certain function. You can open a Request For Enhancement (RFE) at

`https://www.ibm.com/developerworks/support/rational/rfe/`

Log and setup analysis using FEKLOGS

Developer for System z provides a sample job, `FEKLOGS`, which gathers all z/OS UNIX log files as well as Developer for System z installation and configuration information.

Sample job `FEKLOGS` is located in `FEK.#CUST.JCL`, unless you specified a different location when you customized and submitted job `FEK.SFEKSAMP(FEKSETUP)`. See "Customization setup" in the *Host Configuration Guide* (SC23-7658) for more details.

The customization of `FEKLOGS` is described within the JCL. The customization encompasses the provision of a few key variables.

Note: SDSF customers can use the **XDC** line command in SDSF to save the job output in a data set, which in turn can be given to the IBM support center.

Log files

Developer for System z creates log files that can assist you and IBM support center in identifying and solving problems. The following list is an overview of log files that can be created on your z/OS host system. Next to these product-specific logs, be sure to check the SYSLOG for any related messages.

MVS based logs can be located through the appropriate DD statement. z/OS UNIX based log files are located in the following directories:

- userlog/$LOGNAME/

  User-specific log files are located in `userlog/$LOGNAME/`, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rsed.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the `DSTORE_LOG_DIRECTORY` directive is commented out or not present, then `.eclipse/RSE/` is appended to the `user.log` value.
  - `.dstoreMemLogging` - DataStore memory usage logging
  - `.dstoreTrace` - DataStore action logging
  - `fa.log` - The log of the Fault Analyzer Integration
  - `ffs.log` - The log of the Foreign File System (FFS) server, that executes native MVS functions
  - `ffsget.log` - The log of the file reader, that reads a sequential data set or a PDS member
  - `ffsput.log` - The log of the file writer, that writes a sequential data set or a PDS member
  - `lock.log` - The log of the lock manager, that locks/unlocks a sequential data set or a PDS member
  - `rmt_class_loader.cache.jar` - The cache of classes loaded by the RSE remote class loader
  - `rsecomm.log` - The log of the RSE server, that handles commands from the client and the communication logging of all services relying on RSE (may contain Java exception stack trace)
  - `stderr.log` - The redirected data of `stderr`, standard error output
  - `stdout.log` - The redirected data of `stdout`, standard output

  Note: The `.eclipse` directory and the `.dstore*` log files start with a dot (.), which makes them hidden. Use z/OS UNIX command **ls –lA** to list hidden files and directories. When using the Developer for System z client, select the **Window > Preferences... > Remote Systems > Files** preference page and enable "Show hidden files".
- daemon-home

  The RSE daemon and RSE thread pool specific log files are located in `daemon-home`, where `daemon-home` is the value of the `daemon.log` directive in `rsed.envvars`. If the `daemon.log` directive is commented out or not present, the home directory of the user ID assigned to the `RSED` started task is used. The home directory is defined in the OMVS security segment of the user ID.
  - `rsedaemon.log` - The log of the RSE daemon
  - `rseserver.log` - The log of the RSE thread pools

- audit.log - The RSE audit trail
- serverlogs.count - Counter for logging RSE thread pool streams
- stderr.*.log - RSE thread pool standard error stream
- stdout.*.log - RSE thread pool standard output stream

• /tmp

IVP-specific log files (Installation Verification Program) are located in in the directory referenced by TMPDIR, if this variable is defined in rsed.envvars. If the variable is not defined, the files are created in /tmp.

- fekfivpi.log - The log of the fekfivpi IVP test
- fekfivps.log - The log of the fekfivps IVP test
- fekfivpc.log - The communication log of the fekfivpc IVP test

Note: There are operator commands available to control the amount of data written to some of the mentioned log files. Refer to "Operator commands" in the *Host Configuration Guide* (SC23-7658) for more information.

JES Job Monitor logging

• **SYSOUT DD**

Logging of normal operations. The default value in the sample JCL FEK.#CUST.PROCLIB(JMON) is SYSOUT=*.

• **SYSPRINT DD**

Trace logging. The default value in the sample JCL FEK.#CUST.PROCLIB(JMON) is SYSOUT=*. Tracing is activated with the –TV parameter, see "JES Job Monitor tracing" on page 145 for more details.

Lock daemon logging

• **STDOUT DD**

The redirected data of stdout, Java standard output. The default value in the sample JCL FEK.#CUST.PROCLIB(LOCKD) is SYSOUT=*.

• **STDERR DD**

The redirected data of stderr, Java standard error output. The default value in the sample JCL FEK.#CUST.PROCLIB(LOCKD) is SYSOUT=*.

RSE daemon and thread pool logging

• **STDOUT DD**

The redirected data of stdout, Java standard output of RSE daemon. The default value in the sample JCL FEK.#CUST.PROCLIB(RSED) is SYSOUT=*.

• **STDERR DD**

The redirected data of stderr, Java standard error output of RSE daemon. The default value in the sample JCL FEK.#CUST.PROCLIB(RSED) is SYSOUT=*.

• **daemon-home**

The RSE daemon and RSE thread pool specific log files are located in daemon-home, where daemon-home is the value of the daemon.log directive in rsed.envvars. If the daemon.log directive is commented out or not present, the home directory of the user ID assigned to the RSED started task is used. The home directory is defined in the OMVS security segment of the user ID.

- rsedaemon.log - The log of the RSE daemon
- rseserver.log - The log of the RSE thread pools
- audit.log - The RSE audit trail

- `serverlogs.count` - Counter for logging RSE thread pool streams
- `stderr.*.log` - RSE thread pool standard error stream
- `stdout.*.log` - RSE thread pool standard output stream

Note:

- `serverlogs.count`, `stderr.*.log`, and `stdout.*.log` are only created if the `enable.standard.log` directive in `rsed.envvars` is active, or if the function is dynamically activated with the **modify rsestandardlog on** operator command.
- The `*` in `stderr.*.log` and `stdout.*.log` is 1 by default. However, there can be multiple RSE thread pools, in which case the number is incremented for each new RSE thread pool to ensure unique file names.
- There are no user-specific `stdout.log` and `stderr.log` log files when the `enable.standard.log` directive is active. The user-specific data is now written to the matching RSE thread pool stream.
- There are operator commands available to control the amount of data written to some of the mentioned log files. Refer to "Operator commands" in the *Host Configuration Guide* (SC23-7658) for more information.

RSE user logging

- **userlog/$LOGNAME/**

  There are several log files created by the components related to RSE. All are located in `userlog/$LOGNAME/`, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rsed.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the `DSTORE_LOG_DIRECTORY` directive is commented out or not present, then `.eclipse/RSE/` is appended to the `user.log` value.

  - `.dstoreMemLogging` - DataStore memory usage logging
  - `.dstoreTrace` - DataStore action logging
  - `ffs.log` - The log of the Foreign File System (FFS) server, which executes native MVS functions
  - `ffsget.log` - The log of the file reader, that reads a sequential data set or a PDS member
  - `ffsput.log` - The log of the file writer, that writes a sequential data set or a PDS member
  - `lock.log` - The log of the lock manager, that locks or unlocks a sequential data set or a PDS member
  - `rmt_class_loader.cache.jar` - The cache of classes loaded by the RSE remote class loader
  - `rsecomm.log` - The log of the RSE server, that handles commands from the client and the communication logging of all services relying on RSE (may contain Java exception stack trace)
  - `stderr.log` - The redirected data of `stderr`, standard error output
  - `stdout.log` - The redirected data of `stdout`, standard output

Note:

- The `.eclipse` directory and the `.dstore*` log files start with a dot (.), which makes them hidden. Use z/OS UNIX command **ls –lA** to list hidden files and

directories. When using the Developer for System z client, select the **Window > Preferences... > Remote Systems > Files** preference page and enable “Show hidden files”.

- The creation of the `.dstore*` log files is controlled by the `-DDSTORE_*` Java startup options, as described in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC23-7658).
- The `.dstore*` log files are created in ASCII. Use z/OS UNIX command **iconv -f ISO8859-1 -t IBM-1047 .dstore*** to display them in EBCDIC (when using code page IBM-1047).
- There are no user-specific `stdout.log` and `stderr.log` log files when the `enable.standard.log` directive is active. The user-specific data is now written to the matching RSE thread pool stream.
- There are operator commands available to control the amount of data written to some of the mentioned log files. Refer to "Operator commands" in the *Host Configuration Guide* (SC23-7658) for more information.

Fault Analyzer Integration logging

- **userlog/$LOGNAME/**

  Fault Analyzer Integration logging, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rsed.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the `DSTORE_LOG_DIRECTORY` directive is commented out or not present, then `.eclipse/RSE/` is appended to the `user.log` value.

  – `fa.log` - The log of the Fault Analyzer Integration

  – `rsecomm.log` - Communication logging of Fault Analyzer Integration

File Manager Integration logging

- **userlog/$LOGNAME/rsecomm.log**

  Communication logging of File Manager Integration, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rsed.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the `DSTORE_LOG_DIRECTORY` directive is commented out or not present, then `.eclipse/RSE/` is appended to the `user.log` value.

SCLM Developer Toolkit logging

- **userlog/$LOGNAME/rsecomm.log**

  Communication logging of SCLM Developer Toolkit, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rsed.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the `DSTORE_LOG_DIRECTORY` directive is commented out or not present, then `.eclipse/RSE/` is appended to the `user.log` value.

CARMA logging

- **CARMA server job**

When opening a connection with CARMA, using the batch interface, FEK.#CUST.SYSPROC(CRASUBMT) will start a server job (with the user's user ID as owner) named CRAport, where port is the TCP/IP port used.

- **CARMALOG DD**

  If DD statement CARMALOG is specified in the chosen CARMA startup method, CARMA logging is redirected to this DD statement in the server job, otherwise it goes to SYSPRINT.

- **SYSPRINT DD**

  The SYSPRINT DD of the server job holds the CARMA logging, if DD statement CARMALOG is not defined.

- **SYSTSPRT DD**

  The SYSTSPRT DD of the server job holds the system (TSO) messages for the CARMA server startup.

- **userlog/$LOGNAME/rsecomm.log**

  Communication logging of CARMA, where userlog is the combined value of the user.log and DSTORE_LOG_DIRECTORY directives in rsed.envvars, and $LOGNAME is the logon user ID (uppercase). If the user.log directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the DSTORE_LOG_DIRECTORY directive is commented out or not present, then .eclipse/RSE/ is appended to the user.log value.

APPC transaction (TSO Commands service) logging

- **SYSPRINT DD**

  When the APPC administration utility adds and modifies a transaction program (TP) profile, it checks the TP profile and its JCL for syntax errors. Output from this phase consists of TP profile syntax error messages, utility processing messages, and JCL conversion statements. Logging for messages from this phase is controlled by the SYSPRINT DD statement for the ATBSDFMU utility. The default value in sample JCL FEK.SFEKSAMP(FEKAPPCC) is SYSOUT=*. Refer to *MVS Planning: APPC/MVS Management* (SA22-7599) for more details.

- **&SYSUID.FEKFRSRV.&TPDATE.&TPTIME.LOG**

  When a TP executes, the TP runtime messages, such as allocation and termination messages, go to a log named by the MESSAGE_DATA_SET keyword in its TP profile. The default value in sample JCL FEK.SFEKSAMP(FEKAPPCC) is &SYSUID.FEKFRSRV.&TPDATE.&TPTIME.LOG. Refer to *MVS Planning: APPC/MVS Management* (SA22-7599) for more details.

  Note: Depending on your APPC transaction definitions and site defaults, this log file might not appear unless the KEEP_MESSAGE_LOG(ALWAYS) keyword is added to the transaction definitions. Refer to *MVS Planning: APPC/MVS Management* (SA22-7599) for more information about this.

fekfivpc IVP test logging

- **/tmp/fekfivpc.log**

  The fekfivpc command (CARMA related IVP test) will create the fekfivpc.log file to document the communication between RSE and CARMA. The log will be created in the directory referenced by TMPDIR, if this variable is defined in rsed.envvars. If the variable is not defined, the file is created in /tmp.

fekfivpi IVP test logging

- **/tmp/fekfivpi.log**

  Output of the `fekfivpi -file` command (TSO/ISPF Client Gateway related IVP test). The log will be created in the directory referenced by `TMPDIR`, if this variable is defined in `rsed.envvars`. If the variable is not defined, the file is created in `/tmp`.

fekfivps IVP test logging

- **/tmp/fekfivps.log**

  Output of the `fekfivps -file` command (SCLMDT-related IVP test). The log will be created in the directory referenced by `TMPDIR`, if this variable is defined in `rsed.envvars`. If the variable is not defined, the file is created in `/tmp`.

Dump files

When a product abnormally terminates, a storage dump is created to assist in problem determination. The availability and location of these dumps depends heavily on site-specific settings. The dumps may not be created, or the dumps may be created in different locations than those mentioned in the following sections.

MVS dumps

When the program is running in MVS, check the system dump files and check your JCL for the following DD statements (depending on the product):

- SYSABEND
- SYSMDUMP
- SYSUDUMP
- CEEDUMP
- SYSPRINT
- SYSOUT

Refer to *MVS JCL Reference* (SA22-7597) and *Language Environment Debugging Guide* (GA22-7560) for more information on these DD statements.

Java dumps

In z/OS UNIX, most Developer for System z dumps are controlled by the Java Virtual Machine (JVM).

The JVM creates a set of dump agents by default during its initialization (SYSTDUMP and JAVADUMP). You can override this set of dump agents using the `JAVA_DUMP_OPTS` environment variable and further override the set by the use of `-Xdump` on the command line. JVM command-line options are defined in the `_RSE_JAVAOPTS` directive of `rsed.envvars`. Do not change any of the dump settings unless directed by the IBM support center.

Note: The `-Xdump:what` option on the command line can be used for determining which dump agents exist at the completion of startup.

The types of dump that can be produced are the following:

SYSTDUMP
Java Transaction dump. An unformatted storage dump generated by z/OS.

The dump is written to a sequential MVS data set, using a default name of the form `%uid.JVM.TDUMP.%job.D%y%m%d.T%H%M%S`, or as determined by the setting of the `JAVA_DUMP_TDUMP_PATTERN` environment variable. If you do not want transaction dumps to be created, add environment variable `IBM_JAVA_ZOS_TDUMP=NO` to `rsed.envvars`.

Note: `JAVA_DUMP_TDUMP_PATTERN` allows the usage of variables, which are translated to an actual value at the time the transaction dump is taken.

Table 34. JAVA_DUMP_TDUMP_PATTERN variables

Variable	Usage
%uid	User ID
%job	Job name
%y	Year (2 digits)
%m	Month (2 digits)
%d	Day (2 digits)
%H	Hour (2 digits)
%M	Minute (2 digits)
%S	Second (2 digits)

CEEDUMP

Language Environment (LE) dump. A formatted summary system dump that shows stack traces for each thread that is in the JVM process, together with register information and a short dump of storage for each register.

The dump is written to a z/OS UNIX file named `CEEDUMP.yyyymmdd.hhmmss.pid`, where `yyyymmdd` equals the current date, `hhmmss` the current time and `pid` the current process ID. The possible locations of this file are described in "z/OS UNIX dump locations" on page 145.

HEAPDUMP

A formatted dump (a list) of the objects that are on the Java heap.

The dump is written to a z/OS UNIX file named `HEAPDUMP.yyyymmdd.hhmmss.pid.TXT`, where `yyyymmdd` equals the current date, `hhmmss` the current time and `pid` the current process ID. The possible locations of this file are described in "z/OS UNIX dump locations" on page 145.

JAVADUMP

A formatted analysis of the JVM. It contains diagnostic information related to the JVM and the Java application, such as the application environment, threads, native stack, locks, and memory.

The dump is written to a z/OS UNIX file named `JAVADUMP.yyyymmdd.hhmmss.pid.TXT`, where `yyyymmdd` equals the current date, `hhmmss` the current time and `pid` the current process ID. The possible locations of this file are described in "z/OS UNIX dump locations" on page 145.

Refer to *Java Diagnostic Guide* (SC34-6358) for more information on JVM dumps, and *Language Environment Debugging Guide* (GA22-7560) for LE-specific information.

z/OS UNIX dump locations

The JVM checks each of the following locations for existence and write-permission, and stores the CEEDUMP, HEAPDUMP, and JAVADUMP files in the first one available. Note that you must have enough free disk space for the dump file to be written correctly.

1. The directory in environment variable `_CEE_DMPTARG`, if found. This variable is set in `rsed.envvars` as `/tmp`. It can be changed to `/dev/null` to avoid creating the dump files.
2. The current working directory, if the directory is not the root directory (/), and the directory is writable.
3. The directory in environment variable `TMPDIR` (an environment variable that indicates the location of a temporary directory if it is not `/tmp`), if found.
4. The /tmp directory.
5. If the dump cannot be stored in any of the locations mentioned previously, the dump is put in `stderr`.

Tracing

JES Job Monitor tracing

JES Job Monitor tracing is controlled by the system operator, as described in "Operator commands" in the *Host Configuration Guide* (SC23-7658).

- Starting the JMON started task with the `PRM=-TV` parameter activates verbose mode (tracing)
- The **modify –TV** and **modify –TN** commands activate and deactivate tracing

RSE tracing

There are several log files created by the components related to RSE. Most are located in `userlog/$LOGNAME/`, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rsed.envvars`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is commented out or not present, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID. If the `DSTORE_LOG_DIRECTORY` directive is commented out or not present, then `.eclipse/RSE/` is appended to the `user.log` value.

The amount of data written to `ffs*.log`, `lock.log` and `rsecomm.log` is controlled by the **modify rsecommlog** operator command, or by setting `debug_level` in `rsecomm.properties`. See "Operator commands" in the *Host Configuration Guide* (SC23-7658) and "(Optional) RSE tracing" in the *Host Configuration Guide* (SC23-7658) for more details.

The creation of the `.dstore*` log files is controlled by the `–DDSTORE_*` Java startup options, as described in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC23-7658).

Note:

- The `.eclipse` directory and the `.dstore*` log files start with a dot (.), which makes them hidden. Use z/OS UNIX command **ls –lA** to list hidden files and directories. When using the Developer for System z client, select the **Window > Preferences... > Remote Systems > Files** preference page and enable “Show hidden files”.

- The `.dstore*` log files are created in ASCII. Use z/OS UNIX command **iconv -f ISO8859-1 -t IBM-1047 .dstore*** to display them in EBCDIC (when using code page IBM-1047).

The RSE daemon and RSE thread pool specific log files are located in `daemon-home`, where `daemon-home` is the value of the `daemon.log` directive in `rsed.envvars`. If the `daemon.log` directive is commented out or not present, the home directory of the user ID assigned to the `RSED` started task is used. The home directory is defined in the OMVS security segment of the user ID.

The amount of data written to `rsedaemon.log` and `rseserver.log` is controlled by the **modify rsedaemonlog** and **modify rseserverlog** operator commands or by setting `debug_level` in `rsecomm.properties` . See "Operator commands" in the *Host Configuration Guide* (SC23-7658) and "(Optional) RSE tracing" in the *Host Configuration Guide* (SC23-7658) for more details.

`serverlogs.count`, `stderr.*.log`, and `stdout.*.log` are only created if the `enable.standard.log` directive in `rsed.envvars` is active, or if the function is dynamically activated with the **modify rsestandardlog on** operator command..

Lock daemon tracing

The lock daemon-specific log is located in the STDOUT DD of the `LOCKD` started task. The amount of data written to the log is controlled by the `LOG` startup parameter. See "Operator commands" in the *Host Configuration Guide* (SC23-7658) and "(Optional) RSE tracing" in the *Host Configuration Guide* (SC23-7658) for more details.

CARMA tracing

The user can control the amount of trace info CARMA generates by setting Trace Level in the properties tab of the CARMA connection on the client. The choices for `Trace Level` are:

- Disable Logging
- Error Logging
- Warning Logging
- Informational Logging
- Debug Logging

The default value is the following:

```
Error Logging
```

Refer to “Log files” on page 138 for more information on log file locations.

Error feedback tracing

The following procedure allows gathering of information needed to diagnosis error feedback problems with remote build procedures. This tracing will cause performance degradation and should only be done under the direction of the IBM support center. All references to `hlq` in this section refer to the high-level qualifier used during installation of Developer for System z. The installation default is FEK, but this might not apply to your site.

1. Make a backup copy of your active `ELAXFCOC` compile procedure. This procedure is default shipped in data set `hlq.SFEKSAMP`, but could have been copied to a different location, such as `SYS1.PROCLIB`, as described in "ELAXF* remote build procedures" in the *Host Configuration Guide* (SC23-7658).

2. Change the active ELAXFCOC procedure to include the 'MAXTRACE' string on the EXIT(ADEXIT(ELAXMGUX)) compile option.

```
//COBOL  EXEC PGM=IGYCRCTL,REGION=2048K,
//*            PARM=('EXIT(ADEXIT(ELAXMGUX))',
//             PARM=('EXIT(ADEXIT(''MAXTRACE'',ELAXMGUX))',
//             'ADATA',
//             'LIB',
//             'TEST(NONE,SYM,SEP)',
//             'LIST',
//             'FLAG(I,I)'&CICS &DB2 &COMP)
```

   Note: You have to double the apostrophes around MAXTRACE. The option is now: EXIT(ADEXIT(''MAXTRACE'',ELAXMGUX)).

3. Perform a Remote Syntax Check on the COBOL program for which you want detailed tracing.
4. The SYSOUT part of the JES output will start by listing the names of the data sets for SIDEFILE1, SIDEFILE2, SIDEFILE3 and SIDEFILE4.

```
ABOUT TOO OPEN SIDEFILE1 - NAME = 'uid.DT021207.TT110823.M0000045.C0000000'
SUCCESSFUL OPEN SIDEFILE1 - NAME = 'uid.DT021207.TT110823.M0000045.C0000000'
ABOUT TOO OPEN SIDEFILE2 - NAME = 'uid.DT021207.TT110823.M0000111.C0000001'
SUCCESSFUL OPEN SIDEFILE2 - NAME = 'uid.DT021207.TT110823.M0000111.C0000001'
ABOUT TOO OPEN SIDEFILE3 - NAME = 'uid.DT021207.TT110823.M0000174.C0000002'
SUCCESSFUL OPEN SIDEFILE3 - NAME = 'uid.DT021207.TT110823.M0000174.C0000002'
ABOUT TOO OPEN SIDEFILE4 - NAME = 'uid.DT021207.TT110823.M0000236.C0000003'
SUCCESSFUL OPEN SIDEFILE4 - NAME = 'uid.DT021207.TT110823.M0000236.C0000003'
```

   Note: Depending on your settings, SIDEFILE1 and SIDEFILE2 may be pointing to a DD statement (SUCCESSFUL OPEN SIDEFILE1 - NAME = DD:WSEDSF1). Refer to the JESJCL part of the output (which is located before the SYSOUT part) to get the actual data set name.

```
22 //COBOL.WSEDSF1 DD DISP=MOD,
   // DSN=uid.ERRCOB.member.SF.Z682746.XML
23 //COBOL.WSEDSF2 DD DISP=MOD,
   // DSN=uid.ERRCOB.member.SF.Z682747.XML
```

5. Copy these four data sets to your PC, for example, by creating a local COBOL project in Developer for System z and adding the SIDEFILE1->4 data sets.
6. Copy the complete JES job log to your PC, for example, by opening the job output in Developer for System z and saving it to the local project by selecting **File** > **Save As ...** .
7. Restore procedure ELAXFCOC to the original state, either by undoing the change (remove the "MAXTRACE", string in the compile options) or restoring the backup.
8. Send the collected files (SIDEFILE1->4 and job log) to the IBM support center.

z/OS UNIX permission bits

Developer for System z requires that the z/OS UNIX file system and some z/OS UNIX files have certain permission bits set.

SETUID file system attribute

Remote Systems Explorer (RSE) is the Developer for System z component that provides core services such as connecting the client to the host. It must be allowed to perform tasks such as creating the user's security environment.

The file system (HFS or zFS) in which Developer for System z is installed must be mounted with the SETUID permission bit on (this is the system default). Mounting

the file system with the NOSETUID parameter will prevent Developer for System z from creating the user's security environment, and will fail the connection request.

Use the TSO **ISHELL** command to list the current status of the SETUID bit. In the ISHELL panel, select **File_systems > 1. Mount table...** to list the mounted file systems. The **a** line command will show the attributes for the selected file system, where the "Ignore SETUID" field should be 0.

Program Control authorization

Remote Systems Explorer (RSE) is the Developer for System z component that provides core services such as connecting the client to the host. It must run program controlled in order to perform tasks such as switching to the user ID of the client.

The z/OS UNIX program control bit is set during SMP/E install where needed, except for the Java interface to your security product, as documented in Chapter 2, "Security considerations," on page 15. This permission bit might get lost if you did not preserve it during a manual copy of the Developer for System z directories.

The following Developer for System z files must be program controlled:

- /usr/lpp/rdz/bin/
  - fekfdivp
  - fekfomvs
  - fekfrivp
- /usr/lpp/rdz/lib/
  - fekfdir.dll
  - libfekdcore.so
  - libfekfmain.so
- /usr/lpp/rdz/lib/icuc/
  - libicudata.dll
  - libicudata40.1.dll
  - libicudata40.dll
  - libicudata64.40.1.dll
  - libicudata64.40.dll
  - libicudata64.dll
  - libicuuc.dll
  - libicuuc40.1.dll
  - libicuuc40.dll
  - libicuuc64.40.1.dll
  - libicuuc64.40.dll
  - libicuuc64.dll

Use z/OS UNIX command **ls –E** to list the extended attributes, in which the program control bit is marked with the letter p, as shown in the following sample ($ is the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ ls -E lib/fekf*
-rwxr-xr-x  -ps-  2 user     group      94208 Jul  8 12:31 lib/fekfdir.dll
```

Use z/OS UNIX command **extattr +p** to set the program control bit manually, as shown in the following sample ($ and # are the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ su
# extattr +p lib/fekf*
# exit
$ ls -E lib/fekf*
-rwxr-xr-x  -ps-  2 user     group      94208 Jul  8 12:31 lib/fekfdir.dll
```

Note: To be able to use the **extattr +p** command, you must have at least READ access to the `BPX.FILEATTR.PROGCTL` profile in the `FACILITY` class of your security software, or be a superuser (UID 0) if this profile is not defined. For more information, refer to *UNIX System Services Planning* (GA22-7800).

APF authorization

Remote Systems Explorer (RSE) is the Developer for System z component that provides core services such as connecting the client to the host. It must run APF authorized in order to perform tasks such as displaying detailed process resource usage.

The z/OS UNIX APF bit is set during SMP/E install where needed. This permission bit might get lost if you did not preserve it during a manual copy of the Developer for System z directories.

The following Developer for System z files must be APF authorized:

- `/usr/lpp/rdz/bin/`
  - `fekfomvs`
  - `fekfrivp`

Use z/OS UNIX command **ls -E** to list the extended attributes, in which the APF bit is marked with the letter `a`, as shown in the following sample ($ is the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ ls -E bin/fekfrivp
-rwxr-xr-x  aps-  2 user     group     114688 Sep 17 06:41 bin/fekfrivp
```

Use z/OS UNIX command **extattr +a** to set the APF bit manually, as shown in the following sample ($ and # are the z/OS UNIX prompts):

```
$ cd /usr/lpp/rdz
$ su
# extattr +a bin/fekfrivp
# exit
$ ls -E bin/fekfrivp
-rwxr-xr-x  aps-  2 user     group     114688 Sep 17 06:41 bin/fekfrivp
```

Note: To be able to use the **extattr +a** command, you must have at least READ access to the `BPX.FILEATTR.APF` profile in the `FACILITY` class of your security software, or be a superuser (UID 0) if this profile is not defined. For more information, refer to *UNIX System Services Planning* (GA22-7800).

Sticky bit

Some of the optional Developer for System z services require that MVS load modules are available to z/OS UNIX. This is done by creating a stub (a dummy file) in z/OS UNIX with the "sticky" bit on. When the stub is executed, z/OS UNIX will look for an MVS load module with the same name and execute the load module instead.

The z/OS UNIX sticky bit is set during SMP/E install where needed. These permission bits might get lost if you did not preserve them during a manual copy of the Developer for System z directories.

The following Developer for System z files must have the sticky bit on:

- /usr/lpp/rdz/bin/
  - BWBTSOW
  - CRASTART

Use z/OS UNIX command **ls –l** to list the permissions, in which the sticky bit is marked with the letter t, as shown in the following sample ($ is the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ ls -l bin/CRA*
-rwxr-xr-t   2 user     group         71 Jul  8 12:31 bin/CRASTART
```

Use z/OS UNIX command **chmod +t** to set the sticky bit manually, as shown in the following sample ($ and # are the z/OS UNIX prompt):

```
$ cd /usr/lpp/rdz
$ su
# chmod +t bin/CRA*
# exit
$ ls -l bin/CRA*
-rwxr-xr-t   2 user     group         71 Jul  8 12:31 bin/CRASTART
```

Note: To be able to use the **chmod** command, you must have at least READ access to the SUPERUSER.FILESYS.CHANGEPERMS profile in the UNIXPRIV class of your security software, or be a superuser (UID 0) if this profile is not defined. For more information, refer to *UNIX System Services Planning* (GA22-7800).

Reserved TCP/IP ports

With the **netstat** command (TSO or z/OS UNIX) you can get an overview of the ports currently in use. The output of this command will look similar to the following example. The ports used are the last number (behind the "..") in the "Local Socket" column. Since these ports are already in use, they cannot be used for the Developer for System z configuration.

IPv4

```
MVS TCP/IP NETSTAT CS VxRy       TCPIP Name: TCPIP           16:36:42
User Id  Conn     Local Socket          Foreign Socket         State
-------  ----     ------------          --------------         -----
BPXOINIT 00000014 0.0.0.0..10007        0.0.0.0..0             Listen
INETD4   0000004D 0.0.0.0..512          0.0.0.0..0             Listen
RSED     0000004B 0.0.0.0..4035         0.0.0.0..0             Listen
JMON     00000038 0.0.0.0..6715         0.0.0.0..0             Listen
```

IPv6

```
MVS TCP/IP NETSTAT CS VxRy       TCPIP Name: TCPIP           12:46:25
User Id  Conn     State
-------  ----     -----
BPXOINIT 00000018 Listen
  Local Socket:   0.0.0.0..10007
  Foreign Socket: 0.0.0.0..0
INETD4   00000046 Listen
  Local Socket:   0.0.0.0..512
  Foreign Socket: 0.0.0.0..0
RSED     0000004B Listen
```

```
  Local Socket:   0.0.0.0..4035
  Foreign Socket: 0.0.0.0..0
JMON     00000037 Listen
  Local Socket:   0.0.0.0..6715
  Foreign Socket: 0.0.0.0..0
```

Another limitation that can exist is reserved TCP/IP ports. There are the following two common places to reserve TCP/IP ports:

- **PROFILE.TCPIP**

  This is the data set referred to by the PROFILE DD statement of the TCP/IP started task, often named `SYS1.TCPPARMS(TCPPROF)`.

  – PORT: Reserves a port for specified job names.

  – PORTRANGE: Reserves a range of ports for specified job names.

  Refer to *Communications Server: IP Configuration Guide* (SC31-8775) for more information on these statements.

- **SYS1.PARMLIB(BPXPRMxx)**

  – INADDRANYPORT: Specifies the starting port number for the range of port numbers that the system reserves for use with PORT 0, INADDR_ANY binds. This value is only needed for CINET (multiple TCP/IP stacks active on a single host).

  – INADDRANYCOUNT: Specifies the number of ports that the system reserves, starting with the port number specified in the INADDRANYPORT parameter. This value is only needed for CINET (multiple TCP/IP stacks active on a single host).

  Refer to *UNIX System Services Planning* (GA22-7800) and *MVS Initialization and Tuning Reference* (SA22-7592) for more information on these statements.

These reserved ports can be listed with the **`netstat portl`** command (TSO or z/OS UNIX), which creates an output like that in the example as follows:

```
MVS TCP/IP NETSTAT CS VxRy       TCPIP Name: TCPIP           17:08:32
Port# Prot User     Flags    Range       IP Address
----- ---- ----     -----    -----       ----------
00007 TCP  MISCSERV DA
00009 TCP  MISCSERV DA
00019 TCP  MISCSERV DA
00020 TCP  OMVS     D
00021 TCP  FTPD1    DA
00025 TCP  SMTP     DA
00053 TCP  NAMESRV  DA
00080 TCP  OMVS     DA
03500 TCP  OMVS     DAR      03500-03519
03501 TCP  OMVS     DAR      03500-03519
```

Refer to *Communications Server: IP System Administrator's Commands* (SC31-8781) for more information on the **`NETSTAT`** command.

Note: The **`NETSTAT`** command only shows the information defined in `PROFILE.TCPIP`, which should overlap the `BPXPRMxx` definitions. In case of doubt or problems, check the `BPXPRMxx` parmlib member to verify the ports being reserved here.

Address Space size

The RSE daemon, which is a z/OS UNIX Java process, requires a large region size to perform its functions. Therefore it is important to set large storage limits for OMVS address spaces.

Startup JCL requirements

The RSE daemon is started by JCL using BPXBATSL, whose region size must be 0.

Limitations set in SYS1.PARMLIB(BPXPRMxx)

Set MAXASSIZE in SYS1.PARMLIB(BPXPRMxx), which defines the default OMVS address space (process) region size, to 2G. This is the maximum size allowed. This is a system-wide limit, and thus active for all z/OS UNIX address spaces. If this is not desired, then you can set the limit also just for Developer for System z in your security software.

This value can be checked and set dynamically (until the next IPL) with the following console commands, as described in *MVS System Commands* (GC28-1781):

1. DISPLAY OMVS,O
2. SETOMVS MAXASSIZE=2G

Limitations stored in the security profile

Check ASSIZEMAX in the daemon's user ID OMVS segment, and set it to 2147483647 or, preferably, to NONE to use the SYS1.PARMLIB(BPXPRMxx) value.

Using RACF, this value can be checked and set with the following TSO commands, as described in *Security Server RACF Command Language Reference* (SA22-7687):

1. LISTUSER userid NORACF OMVS
2. ALTUSER userid OMVS(NOASSIZEMAX)

Limitations enforced by system exits

Make sure you are not allowing system exits IEFUSI or IEALIMIT to control OMVS address space region sizes. A possible way to accomplish this is by coding SUBSYS(OMVS,NOEXITS) in SYS1.PARMLIB(SMFPRMxx).

SYS1.PARMLIB(SMFPRMxx) values can be checked and activated with the following console commands, as described in *MVS System Commands* (GC28-1781):

1. DISPLAY SMF,O
2. SET SMF=xx

Limitations for 64-bit addressing

Keyword MEMLIMIT in SYS1.PARMLIB(SMFPRMxx) limits how much virtual storage a 64-bit task can allocate above the 2GB bar. Unlike the REGION parameter in JCL, MEMLIMIT=0M means that the process cannot use virtual storage above the bar.

If MEMLIMIT is not specified in SMFPRMxx, the default value is 0M, so tasks are bound to the (31-bit) 2GB below the bar. The default changed in z/OS 1.10 to 2G, allowing 64-bit tasks to use up to 4GB (the 2GB below the bar and the 2GB above the bar granted by MEMLIMIT).

SYS1.PARMLIB(SMFPRMxx) values can be checked and activated with the following console commands, as described in *MVS System Commands* (GC28-1781):

1. DISPLAY SMF,O
2. SET SMF=xx

MEMLIMIT can also be specified as parameter on an EXEC card in JCL. If no MEMLIMIT parameter is specified, the default is the value defined to SMF, except when REGION=0M is specified, in which case the default is NOLIMIT.

Miscellaneous information

Error feedback B37 space abend

When a user selects error feedback during a compile action, several temporary data sets are created by Developer for System z. When one of these data sets runs out of space, the compile jobs ends with a B37-04 space abend.

Adjust the space allocation in FEK.SFEKPROC(FEKFERRF) when your users experience this problem. The default value is SPACE(200,40) TRACKS.

System limits

SYS1.PARMLIB(BPXPRMxx) defines many z/OS UNIX related limitations, which might be reached when several Developer for System z clients are active. Most BPXPRMxx values can be changed dynamically with the **SETOMVS** and **SET OMVS** console commands.

Use the **SETOMVS LIMMSG=ALL** console command to have z/OS UNIX display console messages (BPXI040I) when any of the BPXPRMxx limits is about to be reached.

Connection refused

Each RSE connection starts several processes which are permanently active. New connections can be refused due to the limit set in SYS1.PARMLIB(BPXPRMxx) on the amount of processes, especially when users share the same UID (such as when using the default OMVS segment).

- The limit per UID is set by the MAXPROCUSER keyword and has a default value of 25.
- The system-wide limit is set by the MAXPROCSYS keyword and has a default value of 200.

Another source of refused connections is the limit on the amount of active z/OS address spaces and z/OS UNIX users.

- The maximum amount of Address Space IDs (ASID) is defined in SYS1.PARMLIB(IEASYSxx) with the MAXUSER keyword, and has a default value of 255.
- The maximum amount of z/OS UNIX user IDs (UID) is defined in SYS1.PARMLIB(BPXPRMxx) with the MAXUIDS keyword, and has a default value of 200.

OutOfMemoryError

An RSE thread pool might fail with an OutOfMemoryError message being logged. This error is related to the Java heap size, and might occur if the users active in this thread pool use more resources than anticipated. Common causes of this error are the following things:

- Expanding large data set filters in Remote Systems Explorer
- Opening PDS(E) with a large amount of members
- Opening large members or sequential files

To resolve this issue, you can do the following things:

- Increase the `-Xmx` directive in `rsed.envvars`, because it controls the maximum Java heap size. Note that the Java heap must fit within address space limits.
- Decrease the `-Dmaximum.clients` directive in `rsed.envvars`, because it controls how many users can be placed in a single thread pool (and thus share a single Java heap).

Host Connect Emulator

- Host Connect Emulator uses TN3270 telnet and not the RSE server to connect to the host.
- When using secure telnet (SSL) and you are working with certificates that are not signed by a well-known CA, every client must add the CA certificate to their Host Connect Emulator list of trusted CAs.
- The `NOSNAEXT` option of TCP/IP's `TELNETPARMS` might be necessary to disable the SNA functional extensions. If `NOSNAEXT` is specified, the TN3270 telnet server does not negotiate for contention resolution and SNA sense functions.

Appendix A. Setting up SSL and X.509 authentication

This appendix is provided to assist you with some common problems that you may encounter when setting up Secure Socket Layer (SSL), or during checking or modifying an existing setup. This appendix also provides a sample setup to support users authenticating themselves with an X.509 certificate.

Secure communication means ensuring that your communication partner is who he claims to be, and transmitting information in a manner that makes it difficult for others to intercept and read the data. SSL provides this ability in a TCP/IP network. It works by using digital certificates to identify yourself and a public key protocol to encrypt the communication. Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information on digital certificates and the public key protocol used by SSL.

The actions needed to set up SSL communications for Developer for System z will vary from site to site, depending on the exact needs, the RSE communication method used and what's already available at the site.

In this appendix we will clone the current RSE definitions, so that we have a 2nd RSE daemon connection that will use SSL. We will also create our own security certificates to be used by the different parts of the RSE connection.

Throughout this appendix, a uniform naming convention is used:

- Certificate : `rdzrse`
- Key and certificate storage : `rdzssl.*`
- Password : `rsessl`
- Daemon user ID : `stcrse`

Some tasks described in the following sections expect you to be active in z/OS UNIX. This can be done by issuing the TSO command **OMVS**. Use the **exit** command to return to TSO.

Decide where to store private keys and certificates

The identity certificates and the encryption/decryption keys used by SSL are stored in a key file. Different implementations of this key file exist, depending on the application type.

However, all implementations follow the same principle. A command generates a key pair (a public key and associated private key). The command then wraps the public key into an X.509 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are stored as an entry (identified by an alias) in a key file.

The RSE daemon is a System SSL application and uses a key database file. This key database can be a physical file created by gskkyman or a key ring managed by your SAF-compliant security software (for example, RACF). The RSE server (which is started by the daemon) is a Java SSL application and uses a key store file created by `keytool` or a key ring managed by your security software.

Table 35. SSL certificate storage mechanisms

Certificate storage	Created and managed by	RSE daemon	RSE server
key ring	SAF-compliant security product	supported	supported
key database	z/OS UNIX's gskkyman	supported	/
key store	Java's keytool	/	supported

To connect through SSL, we need both the key store and the key database, either as a z/OS UNIX file or as a SAF-compliant key ring:

- key store (RACF or keytool)
- key database (RACF or gskkyman)

Note:

- SAF-compliant key rings are the preferred method for managing certificates.
- A shared certificate can be used if RSE daemon and RSE server use the same certificate management method.
- RSE daemon must run program controlled. Using System SSL within implies that `SYS1.SIEALNKE` must be made program controlled by your security software.
- In order to run a System SSL application (daemon connection), `SYS1.SIEALNKE` must be in LINKLIST or STEPLIB. If you prefer the STEPLIB method, add the following statement to the end of `rsed.envvars`.

  ```
  STEPLIB=$STEPLIB:SYS1.SIEALNKE
  ```

  Be aware, however, that:

  – Using STEPLIB in z/OS UNIX has a negative performance impact.
  – If one STEPLIB library is APF authorized, then all must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.
- System SSL uses the Integrated Cryptographic Service Facility (ICSF) if it is available. ICSF provides hardware cryptographic support which will be used instead of the System SSL software algorithms. Refer to *System SSL Programming* (SC24-5901) for more information.

Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for information on RACF and digital certificates. `gskkyman` documentation can be found in *System SSL Programming* (SC24-5901), and `keytool` documentation is available at `http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html`.

Create a key ring with RACF

Do not execute this step if you use `gskkyman` to create the RSE daemon key database and `keytool` to create the RSE server key store.

The **RACDCERT** command installs and maintains private keys and certificates in RACF. RACF supports multiple private keys and certificates to be managed as a group. These groups are called key rings.

Refer to *Security Server RACF Command Language Reference* (SA22-7687) for details on the **RACDCERT** command.

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ACCESS(READ) ID(stcrse)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ) ID(stcrse)
SETROPTS RACLIST(FACILITY) REFRESH

RACDCERT ID(stcrse) GENCERT SUBJECTSDN(CN('rdz rse ssl') +
  OU('rdz') O('IBM') L('Raleigh') SP('NC') C('US')) +
  NOTAFTER(DATE(2017-05-21)) WITHLABEL('rdzrse') KEYUSAGE(HANDSHAKE)

RACDCERT ID(stcrse) ADDRING(rdzssl.racf)
RACDCERT ID(stcrse) CONNECT(LABEL('rdzrse') RING(rdzssl.racf) +
  DEFAULT USAGE(PERSONAL))
```

The preceding sample starts by creating the necessary profiles and permitting user ID `STCRSE` access to key rings and certificates owned by that user ID. The user ID used must match the user ID used to run the SSL RSE daemon. The next step is creating a new, self-signed, certificate with label `rdzrse`. No password is needed. This certificate is then added to a newly created key ring (`rdzssl.racf`). Just as with the certificate, no password is needed for the key ring.

The result can be verified with the following `list` option:

```
RACDCERT ID(stcrse) LIST
Digital certificate information for user STCRSE:

 Label: rdzrse
 Certificate ID: 2QjW1OXi0sXZ1aaEqZmihUBA
 Status: TRUST
 Start Date: 2007/05/24 00:00:00
 End Date:   2017/05/21 23:59:59
 Serial Number:
      >00<
 Issuer's Name:
      >CN=rdz rse ssl.OU=rdz.O=IBM.L=Raleigh.SP=NC.C=US<
 Subject's Name:
      >CN=rdz rse ssl.OU=rdz.O=IBM.L=Raleigh.SP=NC.C=US<
 Private Key Type: Non-ICSF
 Private Key Size: 1024
 Ring Associations:
   Ring Owner: STCRSE
   Ring:
      >rdzssl.racf<
```

(Optional) Using a signed certificate

Certificates can be either self-signed or signed by a Certificate Authority (CA). A certificate signed by a CA means that the CA guarantees that the owner of the certificate is who he claims to be. The signing process adds the CA credentials (also a certificate) to your certificate, making it a multi-element certificate chain.

When using a certificate signed by a CA you can avoid trust validation questions by the Developer for System z client, if the client already trusts the CA.

Follow these steps to create and use a CA signed certificate:

1. Create a self-signed certificate.
   ```
   RACDCERT ID(stcrse) GENCERT WITHLABEL('rdzrse') . . .
   ```
2. Create a signing request for this certificate.
   ```
   RACDCERT ID(stcrse) GENREQ (LABEL('rdzrse')) DSN(dsn)
   ```
3. Send the signing request to your CA of choice.
4. Check if the CA credentials (also a certificate) are already known.
   ```
   RACDCERT CERTAUTH LIST
   ```
5. Mark the CA certificate as trusted.
   ```
   RACDCERT CERTAUTH ALTER(LABEL('CA cert')) TRUST
   ```
   Or add the CA certificate to the database.
   ```
   RACDCERT CERTAUTH ADD(dsn) WITHLABEL('CA cert') TRUST
   ```
6. Add the signed certificate to the database; this will replace the self-signed one.
   ```
   RACDCERT ID(stcrse) ADD(dsn) WTIHLABEL('rdzrse') TRUST
   ```

   Note: Do NOT delete the self-signed certificate before replacing it. If you do, you lose the private key that goes with the certificate, which makes the certificate useless.
7. Create a key ring.
   ```
   RACDCERT ID(stcrse) ADDRING(rdzssl.racf)
   ```
8. Add the signed certificate to the key ring.
   ```
   RACDCERT ID(stcrse) CONNECT(ID(stcrse) LABEL('rdzrse')
   RING(rdzssl.racf))
   ```
9. Add the CA certificate to the key ring.
   ```
   RACDCERT ID(stcrse) CONNECT(CERTAUTH LABEL('CA cert')
   RING(rdzssl.racf))
   ```

Note that the CA certificate used to sign your certificate can, in turn, also be signed by another, higher level, CA certificate. If that happens, the higher level CA certificate must also be added to the key ring. This process repeats until the higher level CA certificate is a root CA certificate, which is always a self-signed certificate.

Clone the existing RSE setup

In this step a new instance of the RSE configuration files is created, so that the SSL setup can run parallel with the existing one(s). The following sample commands expect the configuration files to be in `/etc/rdz/`, which is the default location used in "Customization setup" in the *Host Configuration Guide* (SC23-7658).

```
$ cd /etc/rdz
$ mkdir ssl
$ cp rsed.envvars ssl
$ cp ssl.properties ssl
$ ls ssl
rsed.envvars    ssl.properties
```

The z/OS UNIX commands listed in the preceding example create a subdirectory called `ssl` and populate it with the configuration files that require changes. We can share the other configuration files, the installation directory, and the MVS components, because they are not SSL-specific.

By reusing most of the existing configuration files, we can focus on the changes that are actually required for setting up SSL and avoid doing the complete RSE setup again. (For example, we can avoid defining a new location for `ISPF.conf`.)

Update rsed.envvars to enable coexistence

So far, the definitions are an exact copy of the current setup, which implies that the logs of the new RSE daemon will overlay the current server log files. RSE also needs to know where to find the configuration files that were not copied to the `ssl` directory. Both issues can be addressed by minor changes to `rsed.envvars`.

```
$ oedit /etc/rdz/ssl/rsed.envvars
      -> change: _RSE_RSED_PORT=4047
      -> change: -Ddaemon.log=/var/rdz/logs/ssl
      -> change: -Duser.log=/var/rdz/logs/ssl
      -> add at the END:
         # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
         CFG_BASE=/etc/rdz
         CLASSPATH=.:$CFG_BASE:$CLASSPATH
         # --
```

The changes in the preceding example define a new log location (which will be created by RSE daemon if the log location does not exist). The changes also update the `CLASSPATH` so that the SSL RSE processes will first search the current directory (`/etc/rdz/ssl`) for configuration files and then search the original directory (`/etc/rdz`).

Update ssl.properties to enable SSL

By updating `ssl.properties`, RSE is instructed to start using SSL encrypted communication.

```
$ oedit /etc/rdz/ssl/ssl.properties
      -> change: enable_ssl=true
      -> uncomment and change: daemon_keydb_file=rdzssl.racf
      -> uncomment and change: daemon_key_label=rdzrse
      -> uncomment and change: server_keystore_file=rdzssl.racf
      -> uncomment and change: server_keystore_label=rdzrse
      -> uncomment and change: server_keystore_type=JCERACFKS
```

The changes in the preceding example enable SSL and tell the RSE daemon and RSE server that their (shared) certificate is stored under label `rdzrse` in key ring `rdzssl.racf`. The `JCERACFKS` keyword tells RSE server that a SAF-compliant key ring is used as key store.

Note that System SSL (used by the daemon) always uses ICSF, the interface to System z cryptographic hardware, when available. To be able to share the daemon definitions with the server, server_keystore_type `JCECCARACFKS` must be specified. Here, a SAF-compliant key ring is also used as key store for the public keys, but the private key is stored in ICSF. As documented in *Cryptographic Services ICSF Administrator's Guide* (SA22-7521), ICSF uses profiles in the `CSFKEYS` and `CSFSERV` security classes to control who can use cryptographic keys and services.

Activate SSL by creating a new RSE daemon

As stated before, we will create a second connection that will use SSL, which implies creating a new RSE daemon. The RSE daemon can be a started task or user job. We will use the user job method for initial (test) setup. The following instructions expect the sample JCL to be in `FEK.#CUST.PROCLIB(RSED)`, which is the default location used in "Customization setup" in the *Host Configuration Guide* (SC23-7658):

1. Create a new member `FEK.#CUST.PROCLIB(RSEDSSL)` and copy in sample JCL `FEK.#CUST.PROCLIB(RSED)`.
2. Customize `RSEDSSL` by adding a job card on top and an exec statement at the bottom. Also provide the location of the SSL-related configuration files (`/etc/rdz/ssl`), as shown in the following code sample. Note that we enforce the usage of user ID `STCRSE`, as this user ID was given the proper access authority to certificates and key rings in a previous step.

```
//RSEDSSL  JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),USER=STCRSE
//*
//* RSE DAEMON - SSL
//*
//RSED     PROC TMPDIR=,
//            PORT=,
//            IVP=,                    * 'IVP' to do an IVP test
//            CNFG='/etc/rdz/ssl',
//            HOME='/usr/lpp/rdz'
//*
//RSED     EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
// PARM='PGM &HOME./bin/rsed.sh &IVP -C&CNFG -P&PORT -T&TMPDIR'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//          PEND
//*
//RSED     EXEC RSED
//*
```

Figure 26. RSEDSSL - RSE daemon user job for SSL

Note: The user ID assigned to the `RSEDSSL` job must have the same authorizations as the original RSE daemon. `FACILITY` profile `BPX.SERVER` and `PTKTDATA` profile `IRRPTAUTH.FEKAPPL.*` are the key elements here.

Test the connection

The SSL host configuration is complete and the RSE daemon for SSL can be started by submitting job `FEK.#CUST.PROCLIB(RSEDSSL)`, which was created earlier.

The new setup can now be tested by connecting with the Developer for System z client. Since we created a new configuration for use by SSL (by cloning the existing one), a new connection must be defined on the client, using port `4047` for the RSE daemon.

Upon connection, the host and client will start with some handshaking in order to set up a secure path. Part of this handshaking is the exchange of certificates. If the Developer for System z client does not recognize the host certificate or the CA that signed it, Developer for System z client will prompt the user asking if this certificate can be trusted.

Figure 27. Import Host Certificate dialog

By clicking the Finish button the user can accept this certificate as trusted, after which the connection initialization continues.

Note: RSE daemon and RSE server might use two different certificate locations, resulting in two different certificates and thus two confirmations.

Once a certificate is known to the client, this dialog is not shown again. The list of trusted certificates can be managed by selecting **Window** > **Preferences...** > **Remote Systems** > **SSL**, which shows the following dialog:

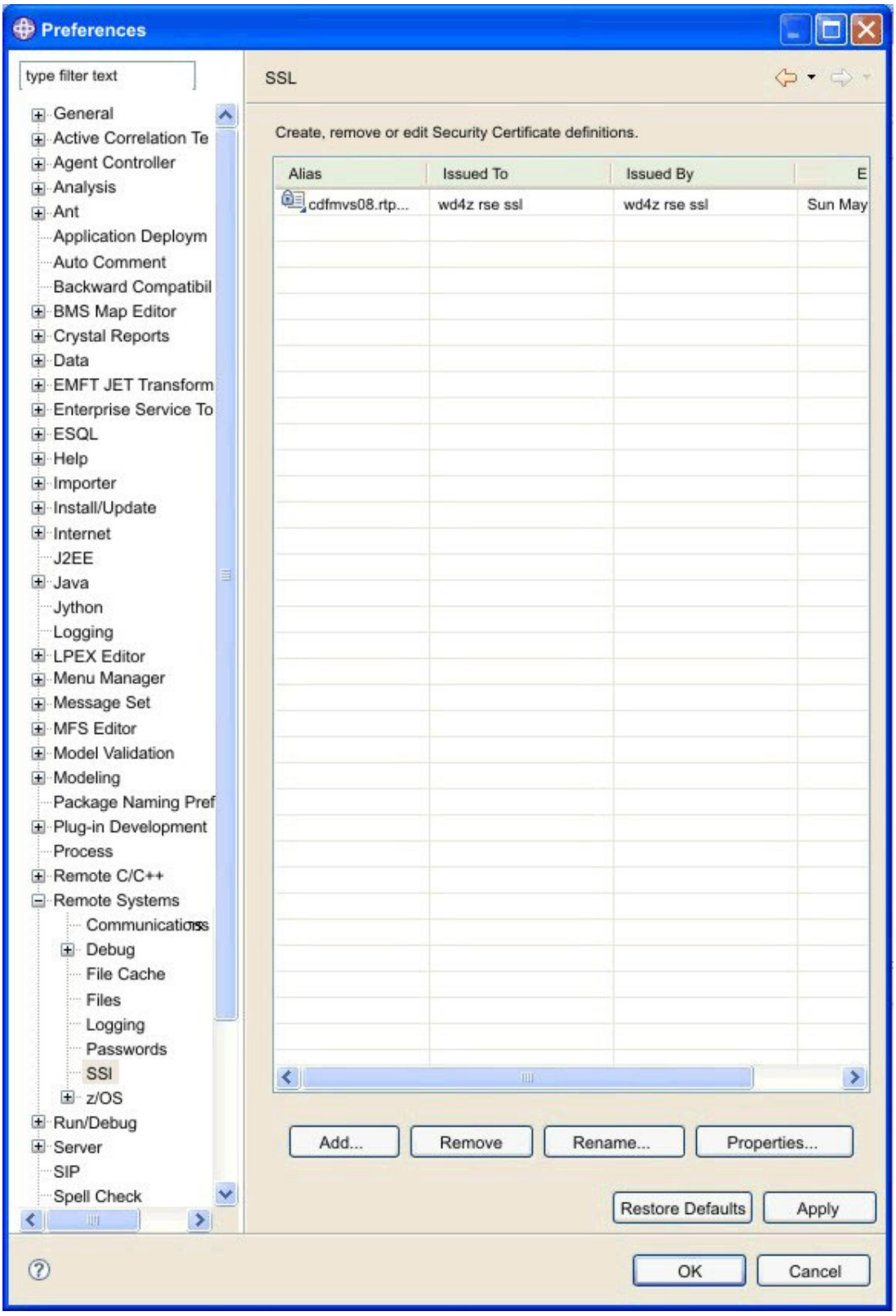


Figure 28. Preferences dialog - SSL

If SSL communication fails, the client will return an error message. More information is available in the different server and user log files, as described in "RSE daemon and thread pool logging" on page 139 and "RSE user logging" on page 140.

(Optional) Add X.509 client authentication support

RSE daemon supports users authenticating themselves with an X.509 certificate. Using SSL encrypted communication is a prerequisite for this function, because it is an extension to the host authentication with a certificate used in SSL.

There are multiple ways to do certificate authentication for a user, as described in "Client authentication using X.509 certificates" on page 24. The next steps document the setup needed to support the method where your security software authenticates the certificate using the HostIdMappings certificate extension.

1. Change the certificate that identifies the Certificate Authority (CA) used to sign the client certificate to a highly trusted CA certificate. Although the TRUST status is sufficient for certificate validation, a change to HIGHTRUST is done, because it is used for the certificate authentication part of the logon process.

   ```
   RACDCERT CERTAUTH ALTER(LABEL('HighTrust CA')) HIGHTRUST
   ```

2. Add the CA certificate to the key ring, `rdzssl.racf`, so that it is available to validate the client certificates.

   ```
   RACDCERT ID(stcrse) CONNECT(CERTAUTH LABEL('HighTrust CA') +
     RING(rdzssl.racf))
   ```

   This concludes the security software setup for the CA certificate.

3. Define a resource (format `IRR.HOST.hostname`) in the `SERVAUTH` class for the host name, `CDFMVS08.RALEIGH.IBM.COM`, defined in the HostIdMappings extension of your client certificate.

   ```
   RDEFINE SERVAUTH  IRR.HOST.CDFMVS08.RALEIGH.IBM.COM  UACC(NONE)
   ```

4. Grant the RSE started task user ID, `STCRSE`, access to this resource with READ authority.

   ```
   PERMIT IRR.HOST.CDFMVS08.RALEIGH.IBM.COM  CLASS(SERVAUTH) +
     ACCESS(READ) ID(stcrse)
   ```

5. Activate your changes to the `SERVAUTH` class. Use the first command if the `SERVAUTH` class is not active yet. Use the second one to refresh an active setup.

   ```
   SETROPTS CLASSACT(SERVAUTH) RACLIST(SERVAUTH)
   or
   SETROPTS RACLIST(SERVAUTH) REFRESH
   ```

   This concludes the security software setup for the HostIdMappings extension.

6. Restart the RSE started task to start accepting client logons using X.509 certificates.

(Optional) Create a key database with gskkyman

Do not execute this step if you use an SAF-compliant key ring for the RSE daemon key database.

`gskkyman` is a z/OS UNIX shell-based, menu-driven, program that creates, populates, and manages a z/OS UNIX file that contains private keys, certificate requests, and certificates. This z/OS UNIX file is called a key database.

Note: The following statements might be necessary to set up the environment for `gskkyman`. Refer to *System SSL Programming* (SC24-5901) for more information about this.

```
PATH=$PATH:/usr/lpp/gskssl/bin
export NLSPATH=/usr/lpp/gskssl/lib/nls/msg/En_US.IBM-1047/%N:$NLSPATH
export STEPLIB=$STEPLIB:SYS1.SIEALNKE
$ cd /etc/rdz/ssl
$ gskkyman       Database Menu

   1 - Create new database

Enter option number: 1
Enter key database name (press ENTER to return to menu): rdzssl.kdb
Enter database password (press ENTER to return to menu): rsessl
Re-enter database password: rsessl
Enter password expiration in days (press ENTER for no expiration):
Enter database record length (press ENTER to use 2500):

Key database /etc/rdz/ssl/rdzssl.kdb created.

Press ENTER to continue.

       Key Management Menu

       6 - Create a self-signed certificate

Enter option number (press ENTER to return to previous menu): 6

       Certificate Type

   5 - User or server certificate with 1024-bit RSA key

Select certificate type (press ENTER to return to menu): 5
Enter label (press ENTER to return to menu): rdzrse
Enter subject name for certificate
  Common name (required): rdz rse ssl
  Organizational unit (optional): rdz
  Organization (required): IBM
  City/Locality (optional): Raleigh
  State/Province (optional): NC
  Country/Region (2 characters - required): US
Enter number of days certificate will be valid (default 365): 3650

Enter 1 to specify subject alternate names or 0 to continue: 0

Please wait .....

Certificate created.

Press ENTER to continue.

       Key Management Menu

         0 - Exit program

Enter option number (press ENTER to return to previous menu): 0
$ ls -l rdzssl.*
total 152
-rw-------   1 IBMUSER  SYS1       35080 May 24 14:24 rdzssl.kdb
-rw-------   1 IBMUSER  SYS1          80 May 24 14:24 rdzssl.rdb
$ chmod 644 rdzssl.*
$ ls -l rdzssl.*
-rw-r--r--   1 IBMUSER  SYS1       35080 May 24 14:24 rdzssl.kdb
-rw-r--r--   1 IBMUSER  SYS1          80 May 24 14:24 rdzssl.rdb
```

The preceding sample starts by creating a key database called `rdzssl.kdb` with password `rsessl`. Once the database exists, it is populated by creating a new, self-signed, certificate, valid for about 10 years (not counting leap days). The

certificate is stored under the label `rdzrse` and with the same password (`rsessl`) as the one used for the key database (this is an RSE requisite).

`gskkyman` allocates the key database with a (very secure) 600 permission bit mask (only owner has access). Unless the daemon uses the same user ID as the creator of the key database, permissions have to be set less restrictive. 644 (owner has read/write, everyone has read) is a usable mask for the **chmod** command.

The result can be verified by selecting the **Show certificate information** option in the **Manage keys and certificates** submenu, as follows:

```
$ gskkyman

       Database Menu

   2 - Open database

Enter option number: 2
Enter key database name (press ENTER to return to menu): rdzssl.kdb
Enter database password (press ENTER to return to menu): rsessl

       Key Management Menu

          1 - Manage keys and certificates

Enter option number (press ENTER to return to previous menu): 1

       Key and Certificate List

          1 - rdzrse

   Enter label number (ENTER to return to selection menu, p for previous list): 1

       Key and Certificate Menu

   1 - Show certificate information

Enter option number (press ENTER to return to previous menu): 1

                        Certificate Information

                Label: rdzrse
            Record ID: 14
      Issuer Record ID: 14
              Trusted: Yes
              Version: 3
         Serial number: 45356379000ac997
           Issuer name: rdz rse ssl
                        rdz
                        IBM
                        Raleigh
                        NC
                        US
          Subject name: rdz rse ssl
                        rdz
                        IBM
                        Raleigh
                        NC
                        US
        Effective date: 2007/05/24
       Expiration date: 2017/05/21
  Public key algorithm: rsaEncryption
       Public key size: 1024
   Signature algorithm: sha1WithRsaEncryption
      Issuer unique ID: None
     Subject unique ID: None
```

```
  Number of extensions: 3

Enter 1 to display extensions, 0 to return to menu: 0

       Key and Certificate Menu

     0 - Exit program

Enter option number (press ENTER to return to previous menu): 0
```

The following `ssl.properties` sample shows that the `daemon_*` directives differ from the SAF key ring sample shown earlier.

```
$ oedit /etc/rdz/ssl/ssl.properties
      -> change: enable_ssl=true
      -> uncomment and change: daemon_keydb_file=rdzssl.kdb
      -> uncomment and change: daemon_keydb_password=rsessl
      -> uncomment and change: daemon_key_label=rdzrse
      -> uncomment and change: server_keystore_file=rdzssl.racf
      -> uncomment and change: server_keystore_label=rdzrse
      -> uncomment and change: server_keystore_type=JCERACFKS
```

The preceding changes enable SSL and tell the RSE daemon that the certificate is stored under label `rdzrse` in key database `rdzssl.kdb` with password `rsessl`. RSE server is still using a SAF compliant key ring.

(Optional) Create a key store with keytool

Do not execute this step if you use a SAF-compliant key ring for the RSE server key store.

"`keytool -genkey`" generates a private key pair and a matching self-signed certificate, which is stored as an entry (identified by an alias) in a (new) key store file.

Note: Java must be included in your command search directories. The following statement might be necessary to be able to execute `keytool`, where `/usr/lpp/java/J5.0` is the directory where Java is installed: `PATH=$PATH:/usr/lpp/java/J5.0/bin`

All information can be passed as a parameter, but due to command-line length limitations some interactivity is required, as follows:

```
$ cd /etc/rdz/ssl
$ keytool -genkey -alias rdzrse -validity 3650 -keystore rdzssl.jks -storepass
 rsessl -keypass rsessl
What is your first and last name?
  [Unknown]:  rdz rse ssl
What is the name of your organizational unit?
  [Unknown]:  rdz
What is the name of your organization?
  [Unknown]:  IBM
What is the name of your City or Locality?
  [Unknown]:  Raleigh
What is the name of your State or Province?
  [Unknown]:  NC
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=rdz rse ssl, OU=rdz, O=IBM, L=Raleigh, ST=NC, C=US correct? (type "yes"
or "no")
  [no]:  yes
$ ls -l rdzssl.*
-rw-r--r--   1 IBMUSER  SYS1        1224 May 24 14:17 rdzssl.jks
```

The self-signed certificate created in the preceding example is valid for about 10 years (not counting leap days). It is stored in `/etc/rdz/ssl/rdzssl.jks` using alias `rdzrse`. Its password (`rsessl`) is identical to the key store password, which is a requisite for RSE.

The result can be verified with the `-list` option, as follows:

```
$ keytool -list -alias rdzrse -keystore rdzssl.jks -storepass rsessl -v
Alias name: rdzrse
Creation date: May 24, 2007
Entry type: keyEntry
Certificate chain length: 1
Certificate 1¨:
Owner: CN=rdz rse ssl, OU=rdz, O=IBM, L=Raleigh, ST=NC, C=US
Issuer: CN=rdz rse ssl, OU=rdz, O=IBM, L=Raleigh, ST=NC, C=US
Serial number: 46562b2b
Valid from: 5/24/07 2:17 PM until: 5/21/17 2:17 PM
Certificate fingerprints:
         MD5:  9D:6D:F1:97:1E:AD:5D:B1:F7:14:16:4D:9B:1D:28:80
         SHA1: B5:E2:31:F5:B0:E8:9D:01:AD:2D:E6:82:4A:E0:B1:5E:12:CB:10:1C
```

The following `ssl.properties` sample shows that the `server_*` directives differ from the SAF key ring sample shown earlier.

```
$ oedit /etc/rdz/ssl/ssl.properties
      -> change: enable_ssl=true
      -> uncomment and change: daemon_keydb_file=rdzssl.racf
      -> uncomment and change: daemon_key_label=rdzrse
      -> uncomment and change: server_keystore_file=rdzssl.jks
      -> uncomment and change: server_keystore_password=rsessl
      -> uncomment and change: server_keystore_label=rdzrse
      -> optionally uncomment and change: server_keystore_type=JKS
```

The preceding changes enable SSL and tell the RSE server that the certificate is stored under label `rdzrse` in key store `rdzssl.jks` with password `rsessl`. RSE daemon is still using a SAF-compliant key ring.

Appendix B. Setting up TCP/IP

This appendix is provided to assist you with some common problems that you may encounter when setting up TCP/IP, or during checking or modifying an existing setup.

Refer to *Communications Server: IP Configuration Guide* (SC31-8775) and *Communications Server: IP Configuration Reference* (SC31-8776) for additional information on TCP/IP configuration.

Hostname dependency

When using APPC for the TSO Commands service, Developer for System z is dependent upon TCP/IP having the correct hostname when it is initialized. This implies that the different TCP/IP and Resolver configuration files must be set up correctly.

You can test your TCP/IP configuration with the `fekfivpt` Installation Verification Program (IVP). The command should return an output like that in this sample ($ is the z/OS UNIX prompt):

```
$ fekfivpt

Wed Jul  2 13:11:54 EDT 2008
uid=1(USERID) gid=0(GROUP)
using /etc/rdz/rsed.envvars

------------------------------------------------------------
TCP/IP resolver configuration (z/OS UNIX search order):
------------------------------------------------------------
Resolver Trace Initialization Complete -> 2008/07/02 13:11:54.745964

res_init Resolver values:
 Global Tcp/Ip Dataset  = None
 Default Tcp/Ip Dataset = None
 Local Tcp/Ip Dataset   = /etc/resolv.conf
 Translation Table      = Default
 UserId/JobName         = USERID
 Caller API             = LE C Sockets
 Caller Mode            = EBCDIC
 (L) DataSetPrefix = TCPIP
 (L) HostName      = CDFMVS08
 (L) TcpIpJobName  = TCPIP
 (L) DomainOrigin  = RALEIGH.IBM.COM
 (L) NameServer    = 9.42.206.2
                     9.42.206.3
 (L) NsPortAddr    = 53            (L) ResolverTimeout    = 10
 (L) ResolveVia    = UDP           (L) ResolverUdpRetries = 1
 (*) Options NDots = 1
 (*) SockNoTestStor
 (*) AlwaysWto     = NO            (L) MessageCase        = MIXED
 (*) LookUp        = DNS LOCAL
res_init Succeeded
res_init Started: 2008/07/02 13:11:54.755363
res_init Ended: 2008/07/02 13:11:54.755371
*********************************************************************
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIP           13:11:54
Tcpip started at 01:28:36 on 06/23/2008 with IPv6 enabled

------------------------------------------------------------
```

```
host IP address:
------------------------------------------------------------
hostName=CDFMVS08
hostAddr=9.42.112.75
bindAddr=9.42.112.75
localAddr=9.42.112.75

Success, addresses match
```

Understanding resolvers

The resolver acts on behalf of programs as a client that accesses name servers for name-to-address or address-to-name resolution. To resolve the query for the requesting program, the resolver can access available name servers, use local definitions (for example, `/etc/resolv.conf`, `/etc/hosts`, `/etc/ipnodes`, `HOSTS.SITEINFO`, `HOSTS.ADDRINFO` or `ETC.IPNODES`), or use a combination of both.

When the resolver address space starts, it reads an optional resolver setup data set pointed to by the SETUP DD card in the resolver JCL procedure. If the setup information is not provided, the resolver uses the applicable native MVS or z/OS UNIX search order without any `GLOBALTCPIPDATA`, `DEFAULTTCPIPDATA`, `GLOBALIPNODES`, `DEFAULTIPNODES` or `COMMONSEARCH` information.

Understanding search orders of configuration information

It is important to understand the search order for configuration files used by TCP/IP functions, and when you can override the default search order with environment variables, JCL, or other variables you provide. This knowledge allows you to accommodate your local data set and HFS file naming standards, and it is helpful to know the configuration data set or HFS file in use when diagnosing problems.

Another important point to note is that when a search order is applied for any configuration file, the search ends with the first file found. Therefore, unexpected results are possible if you place configuration information in a file that never gets found, either because other files exist earlier in the search order, or because the file is not included in the search order chosen by the application.

When searching for configuration files, you can explicitly tell TCP/IP where most configuration files are by using DD statements in the JCL procedures or by setting environment variables. Otherwise, you can let TCP/IP dynamically determine the location of the configuration files, based on search orders documented in *Communications Server: IP Configuration Guide* (SC31-8775).

The TCP/IP stack's configuration component uses TCPIP.DATA during TCP/IP stack initialization to determine the stack's `HOSTNAME`. To get its value, the z/OS UNIX environment search order is used.

Note: Use the trace resolver facility to determine what TCPIP.DATA values are being used by the resolver and where they were read from. For information on dynamically starting the trace, refer to *Communications Server: IP Diagnosis Guide* (GC31-8782). Once the trace is active, issue a TSO **`NETSTAT HOME`** command and a z/OS UNIX shell **`netstat –h`** command to display the values. Issuing a PING of a host name from TSO and from the z/OS UNIX shell also shows activity to any DNS servers that might be configured.

Search orders used in the z/OS UNIX environment

The particular file or table that is searched for can be either an MVS data set or an HFS file, depending on the resolver configuration settings and the presence of given files on the system.

Base resolver configuration files

The base resolver configuration file contains `TCPIP.DATA` statements. In addition to resolver directives, it is referenced to determine, among other things, the data set prefix (`DATASETPREFIX` statement's value) to be used when trying to access some of the configuration files specified in this section.

The search order used to access the base resolver configuration file is the following:

1. **GLOBALTCPIPDATA**

   If defined, the resolver `GLOBALTCPIPDATA` setup statement value is used (see also "Understanding resolvers" on page 170). The search continues for an additional configuration file. The search ends with the next file found.
2. The value of the environment variable **RESOLVER_CONFIG**

   The value of the environment variable is used. This search will fail if the file does not exist or is allocated exclusively elsewhere.
3. **/etc/resolv.conf**
4. **//SYSTCPD DD** card

   The data set allocated to the DD name SYSTCPD is used. In the z/OS UNIX environment, a child process does not have access to the SYSTCPD DD. This is because the SYSTCPD allocation is not inherited from the parent process over the fork() or exec function calls.
5. **userid.TCPIP.DATA**

   `userid` is the user ID that is associated with the current security environment (address space, task, or thread).
6. **jobname.TCPIP.DATA**

   `jobname` is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.
7. **SYS1.TCPPARMS(TCPDATA)**
8. **DEFAULTTCPIPDATA**

   If defined, the resolver `DEFAULTTCPIPDATA` setup statement value is used (see also "Understanding resolvers" on page 170).
9. **TCPIP.TCPIP.DATA**

Translate tables

The translate tables (EBCDIC-to-ASCII and ASCII-to-EBCDIC) are referenced to determine the translate data sets to be used. The search order used to access this configuration file is the following. The search order ends at the first file being found:

1. The value of the environment variable **X_XLATE** The value of the environment variable is the name of the translate table produced by the TSO CONVXLAT command.
2. **userid.STANDARD.TCPXLBIN**

   `userid` is the user ID that is associated with the current security environment (address space or task/thread).

3. **jobname.STANDARD.TCPXLBIN**

   jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.
4. **hlq.STANDARD.TCPXLBIN**

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, hlq is TCPIP by default.
5. If no table is found, the resolver uses a hard-coded default table, identical to the table listed in data set member SEZATCPX(STANDARD).

Local host tables

By default, resolver first attempts to use any configured domain name servers for resolution requests. If the resolution request cannot be satisfied, local host tables are used. Resolver behavior is controlled by TCPIP.DATA statements.

The TCPIP.DATA resolver statements define if and how domain name servers are to be used. The LOOKUP TCPIP.DATA statement can also be used to control how domain name servers and local host tables are used. For more information on TCPIP.DATA statements, refer to *Communications Server: IP Configuration Reference* (SC31-8776).

The resolver uses the Ipv4-unique search order for sitename information unconditionally for getnetbyname API calls. The Ipv4-unique search order for sitename information is the following. The search ends at the first file being found:

1. The value of the environment variable **X_SITE**

   The value of the environment variable is the name of the HOSTS.SITEINFO information file created by the TSO **MAKESITE** command.
2. The value of the environment variable **X_ADDR**

   The value of the environment variable is the name of the HOSTS.ADDRINFO information file created by the TSO **MAKESITE** command.
3. **/etc/hosts**
4. **userid.HOSTS.SITEINFO**

   userid is the user ID that is associated with the current security environment (address space or task/thread).
5. **jobname.HOSTS.SITEINFO**

   jobname is the name specified on the JOB JCL statement for batch jobs or the procedure name for a started procedure.
6. **hlq.HOSTS.SITEINFO**

   hlq represents the value of the DATASETPREFIX statement specified in the base resolver configuration file (if found); otherwise, hlq is TCPIP by default.

Applying this set up information to Developer for System z

As stated before, Developer for System z is dependent upon TCP/IP having the correct hostname when it is initialized, when using APPC. This implies that the different TCP/IP and Resolver configuration files must be set up correctly.

The following example focuses on some configuration tasks for TCP/IP and Resolver. Note that this does not cover a complete setup of TCP/IP or Resolver, it just highlights some key aspects that might be applicable to your site:

1. In the following JCL, you can see that TCP/IP will use SYS1.TCPPARMS(TCPDATA) to determine the stack's hostname.

```
//TCPIP    PROC PARMS='CTRACE(CTIEZB00)',PROF=TCPPROF,DATA=TCPDATA
//*
//* TCP/IP NETWORK
//*
//TCPIP    EXEC PGM=EZBTCPIP,REGION=0M,TIME=1440,PARM=&PARMS
//PROFILE  DD  DISP=SHR,DSN=SYS1.TCPPARMS(&PROF)
//SYSTCPD  DD  DISP=SHR,DSN=SYS1.TCPPARMS(&DATA)
//SYSPRINT DD  SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//ALGPRINT DD  SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CFGPRINT DD  SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD  SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD  SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSERROR DD  SYSOUT=*
```

2. SYS1.TCPPARMS(TCPDATA) tells you that you want the system name to be the hostname and that you do not use a domain name server (DNS); all names will be resolved through site table lookup.

```
; HOSTNAME specifies the TCP host name of this system.  If not
; specified, the default HOSTNAME will be the node name specified
; in the IEFSSNxx PARMLIB member.
;
; HOSTNAME
;
; DOMAINORIGIN specifies the domain origin that will be appended
; to host names passed to the resolver.  If a host name contains
; any dots, then the DOMAINORIGIN will not be appended to the
; host name.
;
DOMAINORIGIN  RALEIGH.IBM.COM
;
; NSINTERADDR specifies the IP address of the name server.
; LOOPBACK (14.0.0.0) specifies your local name server. If a name
; server will not be used, then do not code an NSINTERADDR statement.
; (Comment out the NSINTERADDR line below).  This will cause all names
; to be resolved via site table lookup.
;
; NSINTERADDR  14.0.0.0
;
; TRACE RESOLVER will cause a complete trace of all queries to and
; responses from the name server or site tables to be written to
; the user's console.  This command is for debugging purposes only.
;
; TRACE RESOLVER
```

3. In the Resolver JCL you see that the SETUP DD statement is not used. As mentioned in “Understanding resolvers” on page 170, this means that GLOBALTCPIPDATA and other variables will not be used.

```
//RESOLVER PROC PARMS='CTRACE(CTIRES00)'
//*
//* IP NAME RESOLVER – START WITH SUB=MSTR
//*
//RESOLVER EXEC PGM=EZBREINI,REGION=0M,TIME=1440,PARM=&PARMS
//*SETUP    DD  DISP=SHR,DSN=USER.PROCLIB(RESSETUP),FREE=CLOSE
```

4. If you assume that the RESOLVER_CONFIG environment variable is not set, you can see in Table 36 on page 174 that Resolver will try to use /etc/resolv.conf as base configuration file.

```
TCPIPJOBNAME TCPIP
DomainOrigin RALEIGH.IBM.COM
HostName CDFMVS08
```

As mentioned in “Search orders used in the z/OS UNIX environment” on page 171, the base configuration file contains TCPIP.DATA statements. If the system name is CDFMVS08 (TCPDATA stated that the system name is used as hostname) you can see that /etc/resolv.conf is in sync with SYS1.TCPPARMS(TCPDATA). There are no DNS definitions so site table lookup will be used.

5. Table 36 also tells you that if you do not have to do anything to use the default ASCII-EBCDIC translation table.
6. Assuming that the TSO **MAKESITE** command is not used (can create the X_SITE and X_ADDR variables), /etc/hosts will be the site table used for name lookup.

```
#  Resolver /etc/hosts file cdfmvs08
9.42.112.75   cdfmvs08                    # CDFMVS08 Host
9.42.112.75   cdfmvs08.raleigh.ibm.com    # CDFMVS08 Host
127.0.0.1     localhost
```

The minimal content of this file is information about the current system. In the preceding sample, both cdfmvs08 and cdfmvs08.raleigh.ibm.com are defined as a valid name for the IP address of the z/OS system.

If you were using a domain name server (DNS), the DNS would hold the /etc/hosts info, and /etc/resolv.conf and SYS1.TCPPARMS(TCPDATA) would have statements that identify the DNS to your system.

To avoid confusion, you should keep the TCP/IP and Resolver configuration files in sync with each other.

Table 36. Local definitions available to resolver

File type description	APIs affected	Candidate files
Base resolver configuration files	All APIs	1. GLOBALTCPIPDATA 2. RESOLVER_CONFIG environment variable 3. /etc/resolv.conf 4. SYSTCPD DD-name 5. userid.TCPIP.DATA 6. jobname.TCPIP.DATA 7. SYS1.TCPPARMS(TCPDATA) 8. DEFAULTTCPIPDATA 9. TCPIP.TCPIP.DATA
Translate tables	All APIs	1. X_XLATE environment variable 2. userid.STANDARD.TCPXLBIN 3. jobname.STANDARD.TCPXLBIN 4. hlq.STANDARD.TCPXLBIN 5. Resolver-provided translate table, member STANDARD in SEZATCPX

Table 36. Local definitions available to resolver (continued)

File type description	APIs affected	Candidate files
Local host tables	endhostent endnetent getaddrinfo gethostbyaddr gethostbyname gethostent GetHostNumber GetHostResol GetHostString getnameinfo getnetbyaddr getnetbyname getnetent IsLocalHost Resolve sethostent setnetent	IPv4 1. X_SITE environment variable 2. X_ADDR environment variable 3. /etc/hosts 4. userid.HOSTS.xxxxINFO 5. jobname.HOSTS.xxxxINFO 6. hlq.HOSTS.xxxxINFO IPv6 1. GLOBALIPNODES 2. RESOLVER_IPNODES environment variable 3. userid.ETC.IPNODES 4. jobname.ETC.IPNODES 5. hlq.ETC.IPNODES 6. DEFAULTIPNODES 7. /etc/ipnodes

Note: Table 36 on page 174 is a partial copy from a table in *Communications Server: IP Configuration Guide* (SC31-8775). See that manual for the full table.

Host address is not resolved correctly

When you see problems where TCP/IP Resolver cannot resolve the host address properly, it is most likely due to a missing or incomplete resolver configuration file. A clear indication for this problem is the following message in `lock.log`:

```
clientip(0.0.0.0) <> callerip(<host IP address>)
```

To verify this, execute the `fekfivpt` TCP/IP IVP, as described in "Installation verification" in the *Host Configuration Guide* (SC23-7658). The resolver configuration section of the output will look like the following sample:

```
Resolver Trace Initialization Complete -> 2008/07/02 13:11:54.745964

res_init Resolver values:
 Global Tcp/Ip Dataset  = None
 Default Tcp/Ip Dataset = None
 Local Tcp/Ip Dataset   = /etc/resolv.conf
 Translation Table      = Default
 UserId/JobName         = USERID
 Caller API             = LE C Sockets
 Caller Mode            = EBCDIC
```

Ensure that the definitions in the file (or data set) referenced by "`Local Tcp/Ip Dataset`" are correct.

This field will be blank if you do not use a default name for the IP resolver file (using the z/OS UNIX search order). If so, add the following statement to `rsed.envvars`, where `<resolver file>` or `<resolver data>` represents the name of your IP resolver file:

```
RESOLVER_CONFIG=<resolver file>
```

or

```
RESOLVER_CONFIG='<resolver data set>'
```

Bibliography

Referenced publications

The following publications are referenced in this document:

Table 37. Referenced publications

Publication title	Order number	Reference	Reference Web site
Program Directory for IBM Rational Developer for System z	GI11-8298	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Rational Developer for System z Prerequisites	SC23-7659	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Rational Developer for System z Host Configuration Quick Start	GI11-9201	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Rational Developer for System z Host Configuration Guide	SC23-7658	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Rational Developer for System z Host Configuration Reference	SC14-7290	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Rational Developer for System z Host Configuration Utility Guide	SC14-7282	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Rational Developer for system z Messages and Codes	SC14-7497	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
SCLM Developer Toolkit Administrator's Guide	SC23-9801	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Rational Developer for System z Common Access Repository Manager Developer's Guide	SC23-7660	Developer for System z	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Using APPC to provide TSO command services	SC14-7291	White paper	http://www-306.ibm.com/software/awdtools/rdz/library/
Using ISPF Client Gateway to provide CARMA services	SC14-7292	White paper	http://www-306.ibm.com/software/awdtools/rdz/library/
Communications Server IP Configuration Guide	SC31-8775	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Configuration Reference	SC31-8776	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Diagnosis Guide	GC31-8782	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP System Administrator's Commands	SC31-8781	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/

Table 37. Referenced publications (continued)

Publication title	Order number	Reference	Reference Web site
Communications Server SNA Network Implementation Guide	SC31-8777	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server SNA Operations	SC31-8779	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Cryptographic Services ICSF Administrator's Guide	SA22-7521	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Cryptographic Services System SSL Programming	SC24-5901	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
DFSMS Macro Instructions for Data Sets	SC26-7408	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
DFSMS Using data sets	SC26-7410	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Language Environment Customization	SA22-7564	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Language Environment Debugging Guide	GA22-7560	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Initialization and Tuning Guide	SA22-7591	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Initialization and Tuning Reference	SA22-7592	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS JCL Reference	SA22-7597	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Planning Workload Management	SA22-7602	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Setting Up a Sysplex	SA22-7625	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS System Commands	SA22-7627	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Security Server RACF Command Language Reference	SA22-7687	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Security Server RACF Security Administrator's Guide	SA22-7683	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
TSO/E Customization	SA22-7783	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
TSO/E REXX Reference	SA22-7790	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services Command Reference	SA22-7802	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services Planning	GA22-7800	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services User's Guide	SA22-7801	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Using REXX and z/OS UNIX System Services	SA22-7806	z/OS 1.11	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/

Table 37. Referenced publications (continued)

Publication title	Order number	Reference	Reference Web site
Java Diagnostic Guide	SC34-6650	Java 5.0	http://www.ibm.com/developerworks/java/jdk/diagnosis/
Java SDK and Runtime Environment User Guide	/	Java 5.0	http://www-03.ibm.com/servers/eserver/zseries/software/java/
Resource Definition Guide	SC34-6430	CICSTS 3.1	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html
Resource Definition Guide	SC34-6815	CICSTS 3.2	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html
Resource Definition Guide	SC34-7000	CICSTS 4.1	https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp?topic=/com.ibm.cics.ts.home.doc/library/library_html.html
Resource Definition Guide	SC34-7181	CICSTS 4.2	https://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=/com.ibm.cics.ts.home.doc/library/library_html.html
RACF Security Guide	SC34-6454	CICSTS 3.1	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html
RACF Security Guide	SC34-6835	CICSTS 3.2	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html
RACF Security Guide	SC34-7003	CICSTS 4.1	https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp?topic=/com.ibm.cics.ts.home.doc/library/library_html.html
RACF Security Guide	SC34-7179	CICSTS 4.2	https://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=/com.ibm.cics.ts.home.doc/library/library_html.html
Language Reference	SC27-1408	Enterprise COBOL for z/OS	http://www-03.ibm.com/systems/z/os/zos/bkserv/zapplsbooks.html

The following Web sites are referenced in this document:

Table 38. Referenced Web sites

Description	Reference Web site
Developer for System z Information Center	http://publib.boulder.ibm.com/infocenter/ratdevz/v8r0/index.jsp
Developer for System z Support	http://www.ibm.com/software/rational/products/developer/systemz/
Developer for System z Library	http://www.ibm.com/software/rational/products/developer/systemz/library/index.html
Developer for System z home page	http://www.ibm.com/software/rational/products/developer/systemz/
Developer for System z Recommended service	http://www-01.ibm.com/support/docview.wss?rs=2294&context=SS2QJ2&uid=swg27006335
Developer for System z enhancement request	https://www.ibm.com/developerworks/support/rational/rfe/
z/OS internet library	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
CICSTS Information Center	https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp
IBM Tivoli Directory Server	http://www-01.ibm.com/software/tivoli/products/directory-server/

Table 38. Referenced Web sites (continued)

Description	Reference Web site
Problem Determination Tools Plug-ins	http://www-01.ibm.com/software/awdtools/deployment/pdtplugins/
Download Apache Ant	http://ant.apache.org/
Java keytool documentation	http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html
CA support home page	https://support.ca.com/

Informational publications

The following publications can be helpful in understanding setup issues for requisite host components:

Table 39. Informational publications

Publication title	Order number	Reference	Reference Web site
ABCs of z/OS System Programming Volume 9 (z/OS UNIX)	SG24-6989	Redbook	http://www.redbooks.ibm.com/
System Programmer's Guide to: Workload Manager	SG24-6472	Redbook	http://www.redbooks.ibm.com/
TCPIP Implementation Volume 1: Base Functions, Connectivity, and Routing	SG24-7532	Redbook	http://www.redbooks.ibm.com/
TCPIP Implementation Volume 3: High Availability, Scalability, and Performance	SG24-7534	Redbook	http://www.redbooks.ibm.com/
TCP/IP Implementation Volume 4: Security and Policy-Based Networking	SG24-7535	Redbook	http://www.redbooks.ibm.com/
Tivoli Directory Server for z/OS	SG24-7849	Redbook	http://www.redbooks.ibm.com/

Glossary

Action ID
A numeric identifier for an action between 0 and 999

Application Server
1. A program that handles all application operations between browser-based computers and an organization's back-end business applications or databases. There is a special class of Java-based appservers that conform to the J2EE standard. J2EE code can be easily ported between these appservers. They can support JSPs and servlets for dynamic Web content and EJBs for transactions and database access.
2. The target of a request from a remote application. In the DB2 environment, the application server function is provided by the distributed data facility and is used to access DB2 data from remote applications.
3. A server program in a distributed network that provides the execution environment for an application program.
4. The target of a request from an application requester. The database management system (DBMS) at the application server site provides the requested data.
5. Software that handles communication with the client requesting an asset and queries of the Content Manager.

Bidirectional (bi-di)
Pertaining to scripts such as Arabic and Hebrew that generally run from right to left, except for numbers, which run from left to right. This definition is from the Localization Industry Standards Association (LISA) Glossary.

Bidirectional Attribute
Text type, text orientation, numeric swapping, and symmetric swapping.

Build Request
A request from the client to perform a build transaction.

Build Transaction
A job started on MVS to perform builds after a build request has been received from the client.

Compile
1. In Integrated Language Environment (ILE) languages, to translate source statements into modules that then can be bound into programs or service programs.
2. To translate all or part of a program expressed in a high-level language into a computer program expressed in an intermediate language, an assembly language, or a machine language.

Container
1. In CoOperative Development Environment/400, a system object that contains and organizes source files. An i5/OS® library or an MVS-partitioned data set are examples of a container.
2. In J2EE, an entity that provides life-cycle management, security, deployment, and runtime services to components. (Sun) Each type of container (EJB, Web, JSP, servlet, applet, and application client) also provides component-specific services
3. In Backup Recovery and Media Services, the physical object used to store and move media such as a box, a case, or a rack.
4. In a virtual tape server (VTS), a receptacle in which one or more exported logical volumes (LVOLs) can be stored. A stacked volume containing one or more LVOLs and residing outside a VTS library is considered to be the container for those volumes.
5. A physical storage location of the data. For example, a file, directory, or device.

6. A column or row that is used to arrange the layout of a portlet or other container on a page.
7. An element of the user interface that holds objects. In the folder manager, an object that can contain other folders or documents.

Database
A collection of interrelated or independent data items that are stored together to serve one or more applications.

Data Definition View
Contains a local representation of databases and their objects and provides features to manipulate these objects and export them to a remote database

Data Set
The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

Debug
To detect, diagnose, and eliminate errors in programs.

Debugging Session
The debugging activities that occur between the time that a developer starts a debugger and the time that the developer exits from it.

Error Buffer
A portion of storage used to hold error output information temporarily.

Gateway
1. A middleware component that bridges Internet and intranet environments during Web service invocations.
2. Software that provides services between the endpoints and the rest of the Tivoli environment.
3. A component of a Voice over Internet Protocol that provides a bridge between VoIP and circuit-switched environments.
4. A device or program used to connect networks or systems with different network architectures. The systems may have different characteristics, such as different communication protocols, different network architecture, or different security policies, in which case the gateway performs a translation role as well as a connection role.

Interactive System Productivity Facility (ISPF)
An IBM licensed program that serves as a full-screen editor and dialog manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogs between the application programmer and terminal user. ISPF consists of four major components: DM, PDF, SCLM, and C/S. The DM component is the Dialog Manager, which provides services to dialogs and end-users. The PDF component is the Program Development Facility, which provides services to assist the dialog or application developer. The SCLM component is the Software Configuration Library Manager, which provides services to application developers to manage their application development libraries. The C/S component is the Client/Server, which allows you to run ISPF on programmable workstation, to display the panels using the display function of your workstation operating system, and to integrate workstation tools and data with host tools and data.

Interpreter
A program that translates and runs each instruction of a high-level programming language before it translates and runs the next instruction.

Isomorphic
Each composed element (in other words, an element containing other elements) of the XML instance document starting from

the root has one and only one corresponding COBOL group item whose nesting depth is identical to the nesting depth of its XML equivalent. Each non-composed element (in other words, an element that does not contain other elements) in the XML instance document starting from the top has one and only one corresponding COBOL elementary item whose nesting depth is identical to the nesting level of its XML equivalent and whose memory address at runtime can be uniquely identified.

Linkage Section
The section in the data division of an activated unit (a called program or an invoked method) that describes data items available from the activating unit (a program or a method). These data items can be referred to by both the activated unit and the activating unit.

Load Library
A library containing load modules.

Lock Action
Locks a member.

Navigator View
Provides a hierarchical view of the resources in the Workbench.

Non-Isomorphic
A simple mapping of COBOL items and XML elements belonging to XML documents and COBOL groups that are not identical in shape (non-isomorphic). Non-isomorphic mapping can also be created between non-isomorphic elements of isomorphic structures.

Output Console View
Displays the output of a process and allows you to provide keyboard input to a process.

Output View
Displays messages, parameters, and results that are related to the objects that you work with

Perspective
A group of views that show various aspects of the resources in the workbench. The workbench user can switch perspectives, depending on the task at hand, and customize the layout of views and editors within the perspective.

RAM Repository Access Manager

Remote File System
A file system residing on a separate server or operating system.

Remote System
Any other system in the network with which your system can communicate.

Remote Systems Perspective
Provides an interface for managing remote systems using conventions that are similar to ISPF.

Repository

1. A storage area for data. Every repository has a name and an associated business item type. By default, the name will be the same as the name of the business item. For example, a repository for invoices will be called Invoices. There are two types of information repositories: local (specific to the process) and global (reusable).
2. A VSAM data set on which the states of BTS processes are stored. When a process is not executing under the control of BTS, its state (and the states of its constituent activities) are preserved by being written to a repository data set. The states of all processes of a particular process-type (and of their activity instances) are

stored on the same repository data set. Records for multiple process-types can be written to the same repository.

3. A persistent storage area for source code and other application resources. In a team programming environment, a shared repository enables multiuser access to application resources.
4. A collection of information about the queue managers that are members of a cluster. This information includes queue manager names, their locations, their channels, what queues they host, and so on.

Repository Instance
A project or component that exists in an SCM.

Repositories View
Displays the CVS repository locations that have been added to your Workbench.

Response File

1. A file that contains a set of predefined answers to questions asked by a program and that is used instead of entering those values one at a time.
2. An ASCII file that can be customized with the setup and configuration data that automates an installation. The setup and configuration data would have to be entered during an interactive install, but with a response file, the installation can proceed without any intervention.

Servers View
Displays a list of all your servers and the configurations that are associated with them.

Shell A software interface between users and the operating system that interprets commands and user interactions and communicates them to the operating system. A computer may have several layers of shells for various levels of user interaction.

Shell Name
The name of the shell interface.

Shell Script
A file containing commands that can be interpreted by the shell. The user types the name of the script file at the shell command prompt to make the shell execute the script commands.

Sidedeck
A library that publishes the functions of a DLL program. The entry names and module names are stored in the library after the source code is compiled.

Silent Installation
An installation that does not send messages to the console but instead stores messages and errors in log files. Also, a silent installation can use response files for data input.

Silent Uninstallation
An uninstallation process that does not send messages to the console but instead stores messages and errors in log files after the uninstall command has been invoked.

Task List
A list of procedures that can be executed by a single flow of control.

URL Uniform Resource Locator

Documentation notices for IBM Rational Developer for System z

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
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: 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 states 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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:

Intellectual Property Dept. for Rational Software
IBM Corporation
3039 Cornwallis Road, PO Box 12195
Research Triangle Park, NC 27709
U.S.A.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

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 the names and addresses used by an actual business enterprise 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.

Trademark acknowledgments

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 `www.ibm.com/legal/copytrade.shtml`.

CA Endevor is a registered trademark of CA Technologies.

Rational are trademarks of International Business Machines Corporation and Rational Software Corporation, in the United States, other countries, or both.

Intel and Pentium are trademarks of Intel Corporation in the United States, or other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

Special characters

A

B

C

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Readers' Comments — We'd Like to Hear from You

IBM Rational Developer for System z
Version 8.0.3
Host Configuration Reference Guide

Publication No. SC14-7290-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Send your comments to the address on the reverse side of this form.

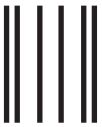
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

Readers' Comments — We'd Like to Hear from You
SC14-7290-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM
Corporation
Building 501
P.O Box 12195
Research Triangle Park, NC
USA 27709-2195

Fold and Tape | **Please do not staple** | Fold and Tape

SC14-7290-01

Cut or Fold
Along Line

Printed in USA

SC14-7290-01