IBM Explorer for z/OS

Host Configuration Reference Guide

SC27-8438-02

IBM Explorer for z/OS

Host Configuration Reference Guide

SC27-8438-02

Note
Before using this information, be sure to read the general information under "Notices" on page 175.

Third edition (September, 2017)

This edition applies to IBM Explorer for z/OS Version 3.1.1 (program number 5655-EX1) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this document

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

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

- *IBM Explorer for z/OS* is called *z/OS Explorer*.
- *z/OS UNIX System Services* is called *z/OS UNIX*.
- *Remote System Explorer* is called *RSE*.
- *IBM Developer for z Systems*™ (previously known as *Rational® Developer for z Systems*) is called *IDz*.

This document is part of a set of documents that describe z/OS Explorer host system configuration. Each of these documents has a specific target audience. You do not have to read all of these documents to complete the z/OS Explorer configuration.

- *IBM Explorer for z/OS Host Configuration Guide* (SC27-8437) describes in detail all of the planning tasks, configuration tasks, and options and provides alternative scenarios.
- *IBM Explorer for z/OS Host Configuration Reference Guide* (SC27-8438) describes z/OS Explorer design and gives background information for various configuration tasks of z/OS Explorer and z/OS components related to z/OS Explorer.
- *IBM Explorer for z/OS Host Configuration Quick Start Guide* (GI13-4313) describes a minimal setup of z/OS Explorer.
- *IBM Explorer for z/OS Host Configuration Utility Guide* (SC27-8436) describes the Host Configuration Utility, an ISPF panel application that guides you through basic and common optional customization steps for z/OS Explorer.

For the most up-to-date versions of the complete documentation, including installation instructions, white papers, podcasts, and tutorials, see the IBM Explorer for z/OS library page.

Who should use this document

This document is intended for system programmers configuring and tuning z/OS Explorer Version 3.0.1.

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 must be familiar with the z/OS UNIX System Services and MVS™ host systems.

Description of the document content

This section summarizes the information presented in this document.

Understanding z/OS Explorer

The z/OS Explorer 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

z/OS Explorer 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

z/OS Explorer 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, z/OS Explorer is not a monolithic application that can be identified easily to Workload Manager (WLM). z/OS Explorer 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 z/OS Explorer. 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 z/OS Explorer setup. However, maintaining hundreds of users, each using multiple threads, a certain amount of storage, and possibly one or more address spaces requires proper configuration of both z/OS Explorer 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 z/OS Explorer.

Push-to-client considerations

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

- Client configuration files
- Client product version

User exit considerations

This chapter assists you with enhancing z/OS Explorer by writing exit routines.

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 z/OS Explorer.

Troubleshooting configuration problems

This chapter is provided to assist you with some common problems that you may encounter during your configuration of z/OS Explorer, 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
- Miscellaneous information
- Host Connect Emulator

Setting up encrypted communication and X.509 authentication

This section is provided to assist you with some common problems that you may encounter when setting up encrypted communication, or during checking or modifying an existing setup. This section also provides a sample setup to support users authenticating themselves with an X.509 certificate.

Setting up TCP/IP

This section 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 z/OS Explorer

The z/OS Explorer 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

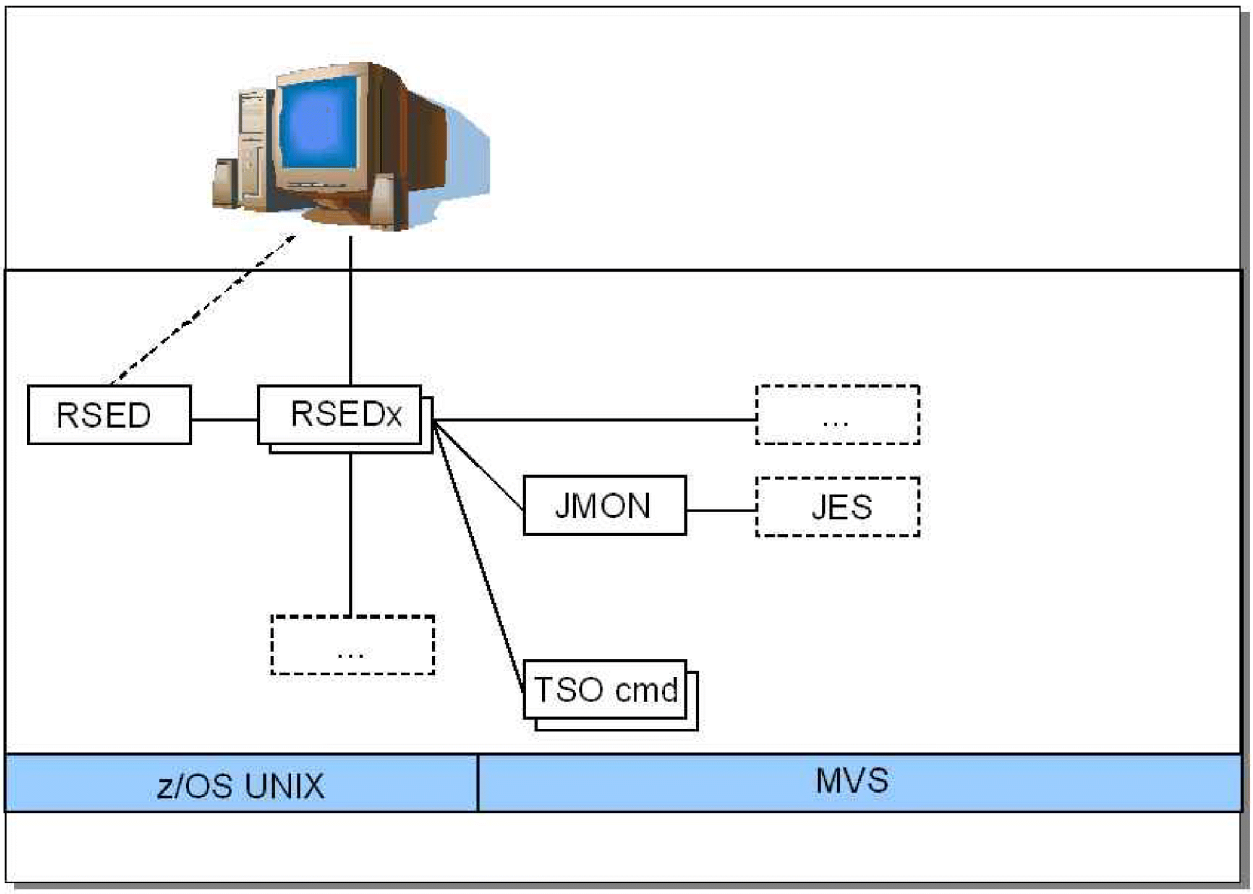


Figure 1. Component overview

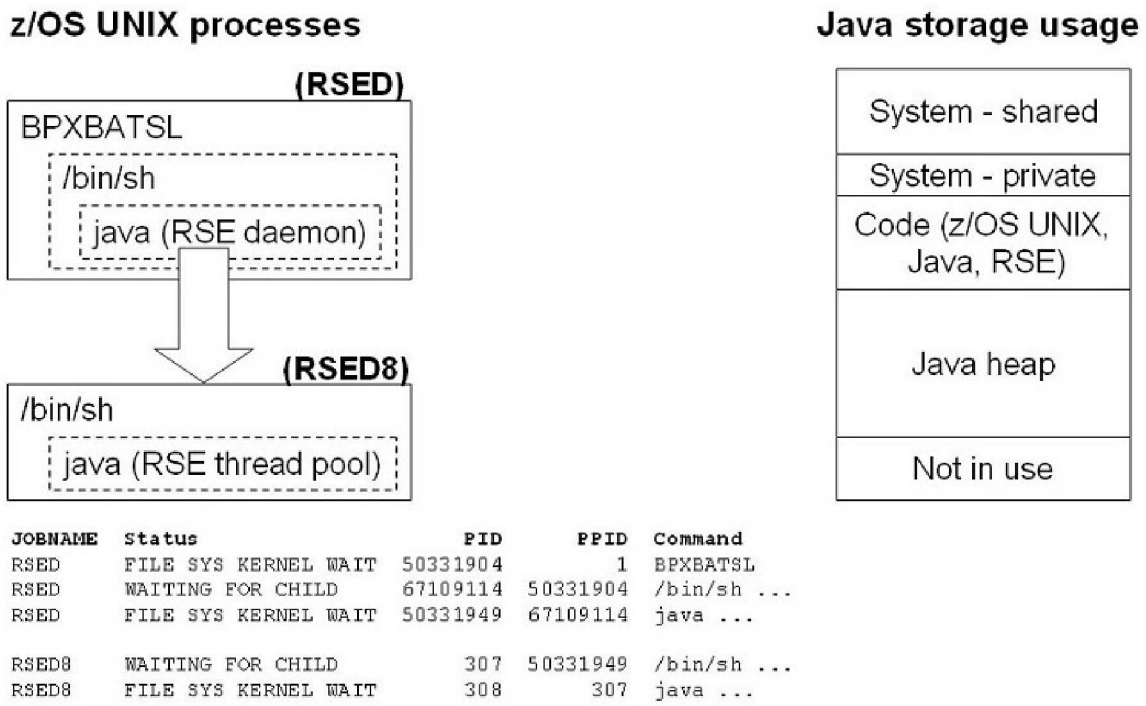
Figure 1 shows a generalized overview of the z/OS Explorer layout on your host system.

- 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.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.
- TSO Commands Service (TSO cmd) provides a interface for TSO and ISPF commands.
- JES Job Monitor (JMON) provides all JES related services.
- More services are available, which can be provided by z/OS Explorer 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 `rse.env` 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 z/OS Explorer 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 z/OS Explorer.

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 on page 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 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 are set and Java is executed with the required parameters to start the RSE thread pool – PID 308.

RSE is capabable of running in 31-bit or 64-bit addressing mode, resulting in different storage limits. In 31-bit mode, the available storage is limited to 2 GB, while in 64-bit mode, there is no limit, unless specified in SYS1.PARMLIB.

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 during Java startup. When running in 64-bit mode, Java will attempt to allocate the heap above the 2 GB bar, freeing up space below the bar.

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

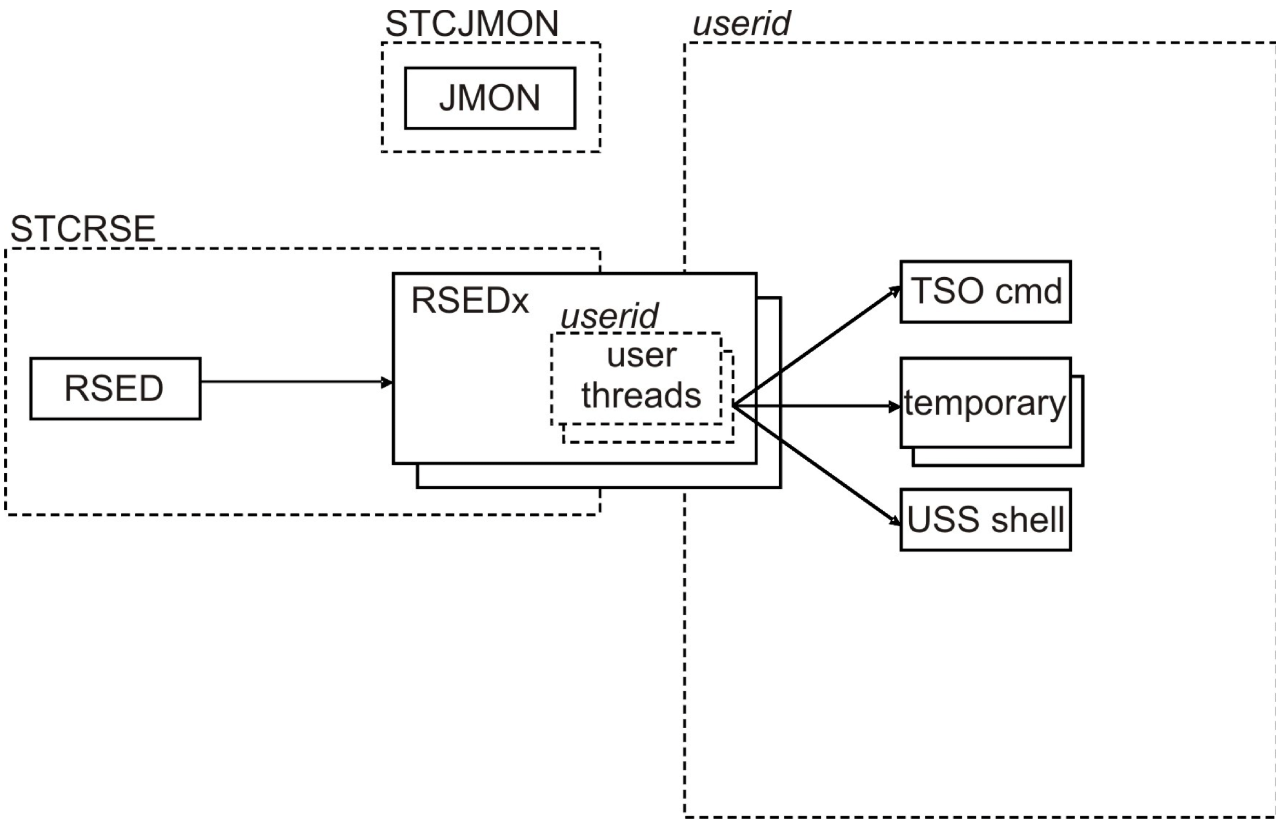


Figure 3. Task owners

Figure 3 shows a basic overview of the owner of the security credentials used for various z/OS Explorer 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 shows the z/OS Explorer started tasks (JMON and RSED), and sample started tasks and system services that z/OS Explorer communicates with.

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.

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

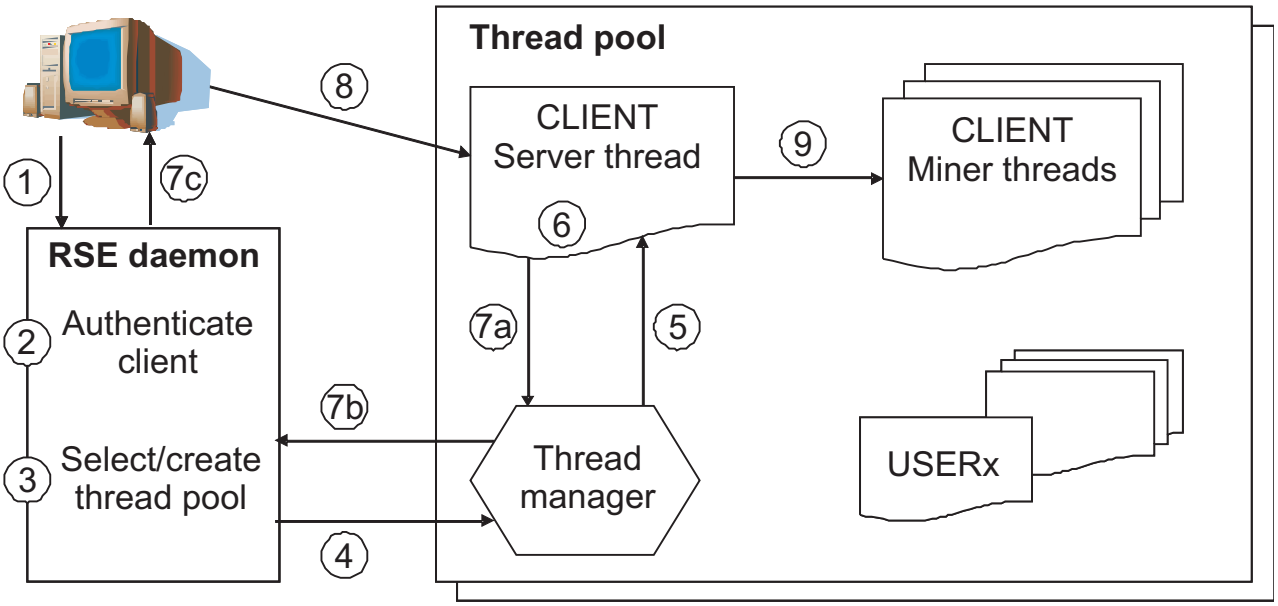


Figure 4. Connection flow

Figure 4 shows a schematic overview of how a client connects to the host using z/OS Explorer. 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.

From a network point of view, z/OS Explorer 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 `rse.env`, then RSE server will bind to a port from this range. If no port is available, the connection fails. 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 z/OS Explorer to invoke these services 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.

Data set lock owner

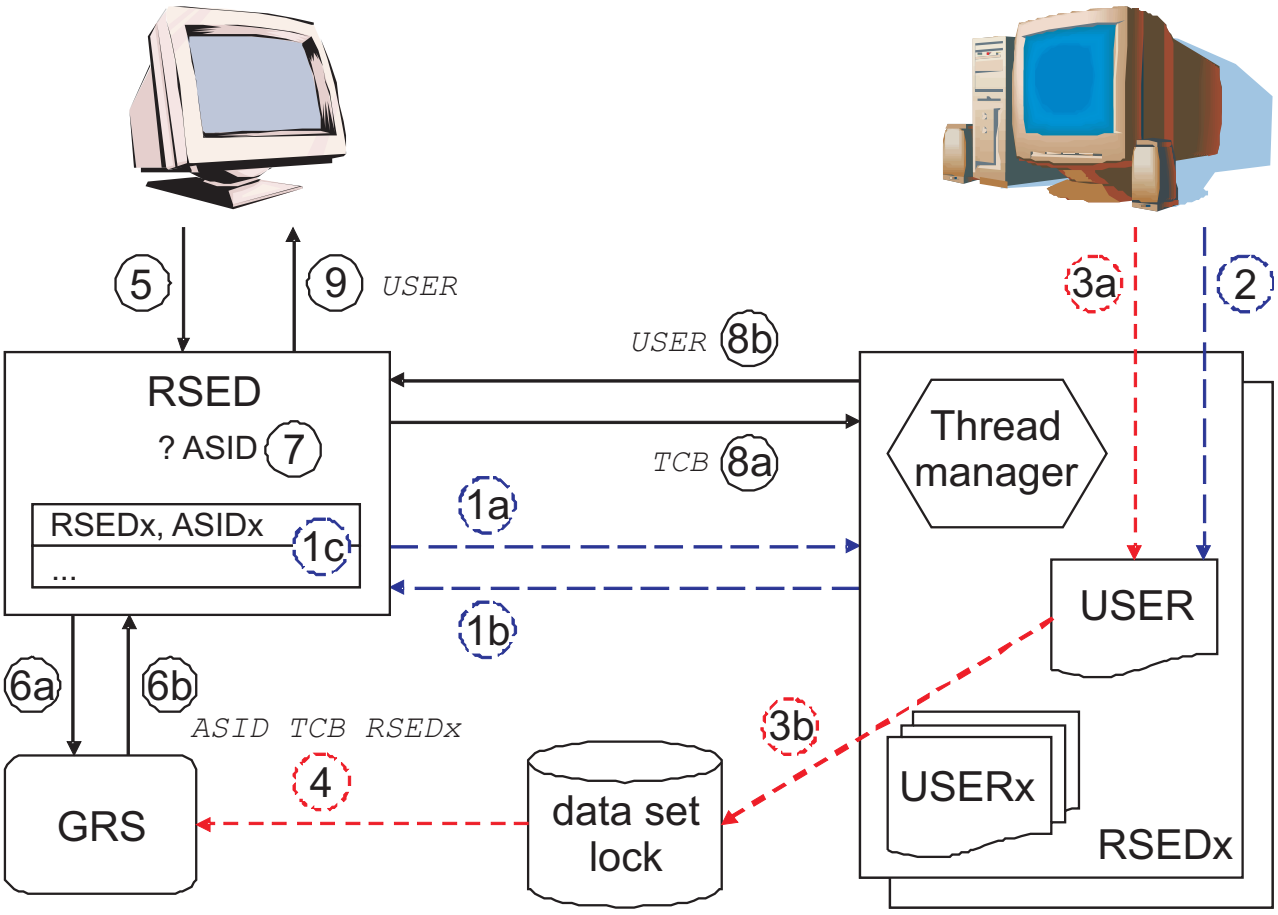


Figure 5. Data set enqueue determination flow

Figure 5 shows a schematic overview of how the RSE daemon determines which z/OS Explorer client owns a data set lock.

1. RSE daemon (RSED) creates a thread pool (RSEDx). To confirm startup completion, the thread pool reports its Address Space Identifier (ASID) back to the RSE daemon, which stores it in the control block created for tracking this thread pool.
2. The client logs on, which creates a user-specific RSE server thread (USER) inside a thread pool (RSEDx). Each thread has a unique Task Control Block (TCB) identifier.
3. The client opens a data set in edit, which instructs RSE server to get an exclusive lock (enqueue) on the data set.
4. The system registers the ASID, TCB and task name (RSEDx) of the requestor as part of the enqueue process. This information is stored in the Global Resource Serialization (GRS) queues.
5. An operator queries the RSE daemon for the lock status of the data set.
6. The RSE daemon scans the GRS queues to learn if the data set is locked and retrieves the ASID, TCB and task name of the lock owner.
7. The retrieved ASID is compared against the ASID of the different thread pools.

8. The RSE daemon asks the thread pool owing the ASID to determine which user owns the TCB.
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 z/OS Explorer, 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 with the **DISPLAY GRS,RES=(*,dataset*)** operator command. System commands stop at address space level, which is the thread pool.

To address this problem, z/OS Explorer provides the **MODIFY rsed APPL=DISPLAY OWNER,DATASET=dataset** operator command, as described in "Operator commands" in the *Host Configuration Guide* (SC27-8437). The operator command can resolve all data set and member locks done by RSE users, as well as locks done by other products, such as ISPF.

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* (SC27-8437). Active data set locks belonging to this user are freed during the process.

z/OS UNIX directory structure

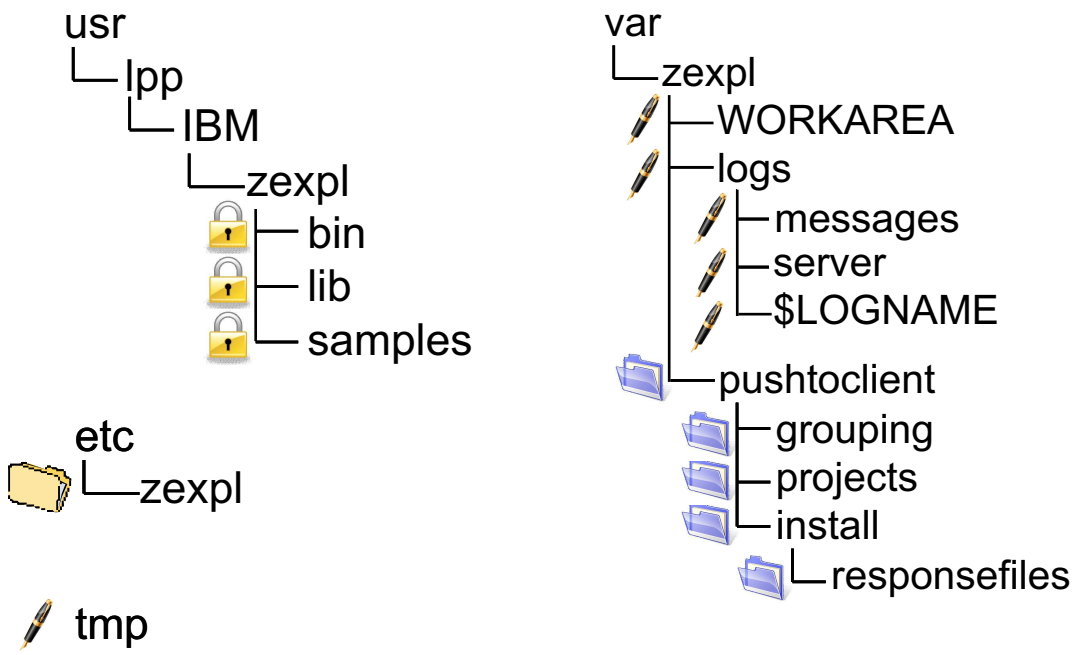


Figure 6. z/OS UNIX directory structure

Figure 6 shows an overview of the z/OS UNIX directories used by z/OS Explorer. The following list describes each directory touched by z/OS Explorer, how the location can be changed, and who maintains the data within.

- `/usr/lpp/IBM/zexpl` is the root path for the z/OS Explorer product code. The actual location is specified in the `RSED` started task (variable `HOME`). The files within are maintained by SMP/E.
- `/etc/zexpl/` holds the RSE and miner-related configuration files. The actual location is specified in the `RSED` started task (variable `CNFG`). The files within are maintained by the system programmer.
- `/tmp/` is used by ISPF 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 `rse.env`. It is also the default location for Java dump files, which can be customized with the `_CEE_DUMPTARG` variable in `rse.env`.

  Note: `/tmp/` requires permission bit mask `777` to allow each client to create temporary files.
- `/var/zexpl/WORKAREA/` is used by ISPF Gateway to transfer data between z/OS UNIX and MVS based address spaces. The actual location is specified in `rse.env` (variable `CGI_ISPWORK`). The files within are maintained by ISPF.

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

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

  Note: /var/zexpl/logs/ requires permission bit mask 777 to allow each client to create his $LOGNAME directory and store the user-specific log files.
- /var/zexpl/pushtoclient/ holds client configuration files, client product update information that is pushed to the client upon connection to the host. The actual location is specified in pushtoclient.properties (variable pushtoclient.folder). The files within are maintained by a z/OS Explorer client administrator.
- /var/zexpl/pushtoclient/grouping/ holds group-specific client configuration files, client product update information that is pushed to the client upon connection to the host. The actual location is specified in pushtoclient.properties (variable pushtoclient.folder). The files within are maintained by a z/OS Explorer client administrator.
- /var/zexpl/pushtoclient/install/ holds configuration files used to update the client product version upon connection to the host. The actual location is specified in /var/zexpl/pushtoclient/keymapping.xml, which is created and maintained by a z/OS Explorer client administrator. The files within are maintained by a client administrator.
- /var/zexpl/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/zexpl/pushtoclient/keymapping.xml, which is created and maintained by a z/OS Explorer client administrator. The files within are maintained by a client administrator.

Update privileges for non-system administrators

The data in /var/zexpl/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/zexpl/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 z/OS Explorer client administrators that maintain the data.

Useful security commands

To ensure that a group of project managers or z/OS Explorer 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 PTCADMIN OMVS(GID(1200))
CONNECT IBMUSER GROUP (PTCADMIN)
ALTUSER IBMUSER DFLTGRP (PTCADMIN)
```

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/zexpl/pushtoclient/
  ```

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

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

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

  ```
  chgrp -R PTCADMIN /var/zexpl/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/zexpl/pushtoclient/
  ```

Sample setup

In the following scenario, all the development project managers, a team of three, are tasked with being a z/OS Explorer client administrator.

The security administrator has already assigned a default group (`PTCADMIN`) 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/zexpl/pushtoclient`.

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

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

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

2. The systems programmer makes `PTCADMIN` the owning group:

   ```
   # chgrp PTCADMIN /var/zexpl/pushtoclient
   # ls –ld /var/zexpl/pushtoclient
   drwxrwxr-x   2 IBMUSER  PTCADMIN /var/zexpl/pushtoclient
   ```

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

Chapter 2. Security considerations

z/OS Explorer 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 z/OS Explorer servers and services rely on the data sets and file systems it resides in being secure. This implies that only trusted system administrators should be able to update the program libraries and configuration files.

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 z/OS Explorer," on page 1 to learn about basic z/OS Explorer design concepts.

Authentication methods

z/OS Explorer supports multiple ways to authenticate a user ID provided by a client upon connection.

- User ID and password
- User ID and one-time password
- User ID and pass phrase
- 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.

User ID and pass phrase

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

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 requires that encrypted communication must be enabled.

JES Job Monitor authentication

Client authentication is done by RSE daemon 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 `rse.env` for RSE and in `FEJJCNFG` for JES Job Monitor.

Connection security

Different levels of communication security are supported by RSE, which controls communication between the client and most z/OS Explorer services:

- External (client-host) communication can be limited to specified ports. This feature is disabled by default.
- External (client-host) communication can be encrypted. 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.

Some optional z/OS Explorer services use a separate external (client-host) communication path:

z/OS Explorer relies on third-party products, such as the TN3270 server, to provide some services. See the related product documentation for connection security options.

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 through the `_RSE_PORTRANGE` definition in `rse.env`.
4. The RSE daemon returns the port number to the client.
5. The client connects to the host port.

Communication encryption

All external z/OS Explorer data streams that pass through RSE can be encrypted. The encryption details are controlled by `GSK_*` variables in `rse.env` and the settings in the `ssl.properties` configuration file, as described in Encrypted communication.

Port Of Entry checking

z/OS Explorer 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 `rse.env`, 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 (security token) 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 `rse.env` 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 z/OS Explorer during logon, as multiple PassTickets will be required within a second. Therefore, z/OS Explorer 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

z/OS Explorer 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 `rse.env` influence the audit function, as documented in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC27-8437).

- 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 complete path to the audit logs is `daemonlog/server`, where `daemonlog` is the value of the `daemon.log` option.

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* (SC27-8437).

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.

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 `rse.env` 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 `rse.env`. 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)
- File access (read, write, create, delete, rename)
- Execution of TSO and z/OS UNIX 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. For example:

```
yyyy/mm/dd hh:mm:ss.sss,userid,action,dataset_name[,returncode]
  [,additional_information]]
```

Data set and member statistics are also logged when the file is opened. They are appended to the line documenting completion of the READ action, and the fields are delimited with %n. For example:

```
yyyy/mm/dd hh:mm:ss.sss,userid,action,dataset_name,returncode,create%nmodify%n...
```

The following attributes are logged, in the listed order:

- Creation date and time (mm/dd/yyyy hh:mm)
- Last modify date and time (mm/dd/yyyy hh:mm:ss)
- Last access date and time (mm/dd/yyyy hh:mm:ss)
- Record format (RECFM)
- SCLM revision indicator (N = revision number is set, D = revision number is not set)
- SCLM revision number
- "Bad Hex" characters included (Y = yes, N = no)

  Note: "Bad Hex" characters require z/OS Explorer mapping services because they do not survive a trip to the client and back due to code page mismatches.
- Logical record length (LRECL)
- File size
- Reserved for future use
- Reserved for future use
- User ID
- Lock (enqueue) owner for this data set or member
- CR (carriage return), LF (line feed) and NL (new line) host code points and their substitution characters.

JES security

z/OS Explorer 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 z/OS Explorer 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. This table lists 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* (SC27-8437).

Table 2. LIMIT_COMMANDS command permission matrix. This table lists `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. This table lists 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

Table 3. Extended JESSPOOL profiles (continued). This table lists extended JESSPOOL profiles.

Command	JESSPOOL profile	Required access
Show JCL	nodeid.userid.jobname.jobid.JCL	READ

Use the following substitutions in the preceding table.

Table 4. Substitutions. This table lists substitutions for the names used in the preceding table.

Name	Substitution
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* (SC27-8437). 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.

As documented in the *JES2 Initialization and Tuning Guide (SA22-7532)*, and the *JES3 Initialization and Tuning Guide (SA22-7549)*, users require UPDATE access to the related OPERCMDS profiles.

Table 5. OPERCMDS profiles checked by JES.

Action	JES2	JES3
Hold	jesname.MODIFYHOLD.BAT jesname.MODIFYHOLD.STC jesname.MODIFYHOLD.TSU	jesname.MODIFY.JOB
Release	jesname.MODIFYRELEASE.BAT jesname.MODIFYRELEASE.STC jesname.MODIFYRELEASE.TSU	jesname.MODIFY.JOB
Cancel	jesname.CANCEL.BAT jesname.CANCEL.STC jesname.CANCEL.TSU	jesname.MODIFY.JOB
Purge	jesname.CANCEL.BAT jesname.CANCEL.STC jesname.CANCEL.TSU	jesname.MODIFY.JOB
Show JCL	not applicable	not applicable

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.

Note: Even if you are not authorized for these operator commands, you can still submit jobs and read job output through JES Job Monitor if you have authority to possible profiles that protect these resources, such as those in the JESINPUT, JESJOBS, and JESSPOOL classes.

For more information about operator command protection, see *Security Server RACF Security Administrator's Guide* (SA22-7683).

Actions against jobs - console

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* (SC27-8437). The default console name is `JMON`.

JES Job Monitor allows you to define how much authority is granted to the EMCS console with the `LIMIT_CONSOLE` directive, as documented in "FEJJCNFG, JES Job Monitor configuration file" in the *Host Configuration Guide* (SC27-8437).

Table 6. LIMIT_CONSOLE console authority matrix

LIMIT_CONSOLE	Active profile in OPERCMDS class	No active profile in OPERCMDS class
LIMITED (default)	Allowed, if permitted by security profile	Not allowed
NOLIMIT	Allowed, if permitted by security profile	Allowed

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 if `LIMIT_CONSOLE=LIMITED` is defined in `FEJJCNFG`.
- 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.

When JES Job Monitor creates an EMCS console for a user, console message IEA630I is written to the system log. In this message, `LU=userid` identifies for which user the console is created. So even when you use a fixed console name such as the default JMON, you can still see which user issued the command.

```
stcjobid IEA630I OPERATOR console NOW ACTIVE,   SYSTEM=sysid, LU=userid
console  jes_command
console  jes_response
stcjobid IEA631I OPERATOR console NOW INACTIVE, SYSTEM=sysid, LU=userid
```

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 an EMCS console, 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).

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* (SC27-8437).

Table 7. 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.

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

Encrypted communication

Note: Due to a vulnerability in the SSLv3 (Secure Socket Layer) protocol, support for this protocol is deprecated in z/OS Explorer.

External (client-host) communication using RSE can be encrypted using Transport Layer Security (TLS). This feature is disabled by default and is controlled by the settings in ssl.properties. Refer to "(Optional) ssl.properties, RSE encrypted communications" in the *Host Configuration Guide* (SC27-8437).

RSE daemon and RSE server support different mechanisms to store certificates due to architectural differences between the two. SAF-compliant key rings are supported by both RSE daemon and RSE server, and 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 z Systems 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 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 `rse.env`.

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`

You can use various variables that are documented in *rse.env, the RSE configuration file* chapter in the *Host Configuration Guide (SC27-8437)* to control the encrypted communication:

- Use the `GSK_PROTOCOL_*` variables in `rse.env` to control which protocols can be used for encrypted communication.
- Use the `GSK_V3_CIPHERS` variable in `rse.env` to control which group of ciphers can be used for encrypted communication.
- Use the `GSK_V3_CIPHER_SPECS` and `GSK_V3_ CIPHER_SPECS_EXPANDED` variables in `rse.env` to control which ciphers can be used for encrypted communication.
- Use the `GSK_FIPS_STATE` variable in `rse.env` to enforce FIPS 140-2 compliant encrypted communication.

Note:

- z/OS Explorer will disable ciphers that are known to be insecure.
- Implicit and explicit protocol and cipher selections are propagated by RSE daemon to RSE server.

See Chapter 12, "Setting up encrypted communication and X.509 authentication," on page 145 for more details on activating encrypted communications for z/OS Explorer.

Note: The z/OS Explorer client and host must have access to common encryption protocols and common cipher suite definitions to be able to set up encrypted communication. For information on Java cipher suite definitions used by the client and RSE server, see the developerWorks® Java technology security information site (http://www.ibm.com/developerworks/java/jdk/security/). For information on System SSL cipher suite definitions used by RSE daemon, see *Cryptographic Services System SSL Programming (SC24-5901)*.

Client authentication using X.509 certificates

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

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 `rse.env` 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 z/OS Explorer.

If you are familiar with the 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 an SAF key ring, 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" 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(keyring.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. 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 `rse.env`.

- 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 `rse.env`, as they might change the way RSE daemon handles encrypted 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 z/OS Explorer, as these filters map all certificates to a single user ID. The result is that all your z/OS Explorer 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

z/OS Explorer 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 `rse.env` 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 25.

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 24 for more information on accessible CA certificates.

Port Of Entry (POE) checking

z/OS Explorer 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 `rse.env`, as documented in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC27-8437).
- 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.

Altering client functions

z/OS Explorer clients check access authorization to SAF security profiles, and based on the result enable or disable the related function for the user.

z/OS Explorer verifies access permits to the profiles listed in Table 8 on page 28 to determine which options should enabled or disabled for the user.

Table 8. SAF information for altering client functions

FACILITY profile	Fixed length	Required access	Result
FEK.USR.OFF.REMOTECOPY.MVS.sysname	27	READ	Client disables copy and related functions for MVS data sets

Note: z/OS Explorer 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 `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, z/OS Explorer expects the `FEK.*` 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.USR.<key>`) and the length of the site-specific profile part (`sysname`) exceeds this number, you can place the profiles in another class and instruct z/OS Explorer to use this class instead. To do so, uncomment `_RSE_FEK_SAF_CLASS` in `rse.env` and provide the desired class name, for example XFACILIT.

The following sample security definitions allow the REMOTECOPY.MVS action for all users on CDFMVS08, except those in group RESTRICT:

```
RDEFINE FACILITY (FEK.USR.OFF.REMOTECOPY.MVS.CDFMVS08) -
  UACC(NONE) DATA('IBM Explorer for z/OS - CLIENT CONTROL')
PERMIT FEK.USR.OFF.REMOTECOPY.MVS.CDFMVS08 CLASS(FACILITY) -
  ID(RESTRICT) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

OFF.REMOTECOPY.MVS

When users have READ access to profile `FEK.USR.OFF.REMOTECOPY.MVS.sysname`, then their z/OS Explorer clients will disable drag, copy, save-as, and work-offline actions for MVS data sets. The result is that the users can access the data sets on this system, but the users cannot create a local copy of a data set on their workstation. This helps prevent exposure of confidential information if the local workstation is lost or stolen.

Push-to-client developer groups

z/OS Explorer clients 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.

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`), z/OS Explorer verifies access permits to the profiles listed in Table 9 to determine which developer groups the user belongs to, and whether a user is allowed to reject updates.

Table 9. 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[.devgroup]	30	READ	User can reject configuration updates
FEK.PTC.REJECT.PRODUCT. UPDATES.sysname[.devgroup]	31	READ	User can reject product updates

Note: z/OS Explorer 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 z/OS Explorer 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, z/OS Explorer expects the `FEK.*` 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 z/OS Explorer to use this class instead. To do so, uncomment `_RSE_FEK_SAF_CLASS` in `rse.env` and provide the desired class name, for example XFACILIT.

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* (SC27-8437) for more information about enabling multiple group support. See Chapter 7, "Push-to-client considerations," on page 97 for more information about push-to-client concepts and implementation.

Send message security

z/OS Explorer allows sending messages to clients, similar to the TSO **SEND** command. Messages can be sent via an operator command (**MODIFY SEND**), a z/OS UNIX command (**send**) or the TSO **SEND** command. Messages are delivered immediately to connected users, and are also stored for users that log on later. This message buffer can be cleared using the operator command and the z/OS UNIX command.

z/OS Explorer will query your security product for access permits to `FEK.CMD.SEND.**` profiles to determine if the requestor is allowed to send a message or clear the buffer. The RSED started task user ID is verified when using the operator command interface.

Table 10. Send message SAF information

FACILITY profile	Fixed length	Required access	Result
FEK.CMD.SEND.CLEAR.jobname	19	READ	The requestor can clear the message buffer of jobname
FEK.CMD.SEND.MSG.jobname	17	READ	The requestor can send messages to users of jobname

Note: z/OS Explorer 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 `jobname` value matches the RSED started task name.

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

By default, z/OS Explorer expects the `FEK.*` 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.CMD.SEND.<key>`) and the length of the site-specific profile part (`jobname`) exceeds this number you can place the profiles in another class and instruct z/OS Explorer to use this class instead. To do so, uncomment `_RSE_FEK_SAF_CLASS` in `rse.env` and provide the desired class name, for example XFACILIT.

Access violations are reported with console message FEK302E.

The following sample security definitions allow everyone to send messages, but only users able to issue operator commands can clear the message buffer.

```
RDEFINE FACILITY (FEK.CMD.SEND.**) UACC(READ) -
  DATA('z/OS EXPLORER - SEND COMMAND')
RDEFINE FACILITY (FEK.CMD.SEND.CLEAR.**) UACC(NONE) -
  DATA('z/OS EXPLORER - CLEAR SEND BUFFER')
PERMIT FEK.CMD.SEND.CLEAR.** CLASS(FACILITY) -
  ID(STCRSE) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Log file security

Log creation

The log directories and log files created by z/OS Explorer have, by default, secure access permissions where only the owner has access (read and write). For server (and audit) logs, the owner is the RSED started task user ID. For user logs the owner is the user ID provided by the end user during logon. The `log.file.mode` directive in `rse.env` can be used to set different access permissions. Note that the access permissions for audit files are controlled separately, and are set with the `audit.log.mode` directive in `rse.env`.

Before writing to a log directory, z/OS Explorer will validate the file ownership, and will fail the write if a different user owns the file. The `log.secure.mode` directive in `rse.env` can be used to disable the ownership check.

Log collection – requirements for requestor

The RSED start task supports the **MODIFY LOGS** operator command to collect z/OS Explorer host logs and setup information. The collected data is placed in a z/OS UNIX file, `$TMPDIR/feklogs%sysname.%jobname`, where `$TMPDIR` is the value of the `TMPDIR` directive in `rse.env` (default `/tmp`), `%sysname` is your z/OS system name and `%jobname` is the name of the RSED started task.

The z/OS Explorer client can also request the RSED started task to collect host logs and setup information and send it back to the client.

z/OS Explorer will query your security product for access permits to `FEK.CMD.LOGS.**` profiles to determine if the requestor is allowed to collect the specified logs. By default, the requestor is the RSED started task user ID, unless the `OWNER` option is specified. Only the requestor has access to the file holding the collected data.

FACILITY profile	Fixed length	Required access	Result
FEK.CMD.LOGS.AUDIT.jobname	19	READ	The requestor can collect audit logs of jobname
FEK.CMD.LOGS.SERVER.jobname	20	READ	The requestor can collect server logs of jobname
FEK.CMD.LOGS.USER.userid	18	READ	The requestor can collect user logs of userid
FEK.CMD.LOGS.OWNER.userid	19	READ	The requestor is changed from RSED started task user ID to userid

Note: z/OS Explorer assumes that a user has 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 `jobname` value matches the RSED started task name. The `userid` value matches a valid user ID.

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

By default, z/OS Explorer expects the FEK.* 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.CMD.LOGS.<key>) and the length of the site-specific profile part (jobname or userid) exceeds this number you can place the profiles in another class and instruct z/OS Explorer to use this class instead. To do so, uncomment _RSE_FEK_SAF_CLASS in rse.env and provide the desired class name, for example XFACILIT.

Access violations are reported with console message FEK302E.

The following sample security definitions allow everyone to collect host logs, but only group SYSPROG can collect audit data:

```
RDEFINE FACILITY (FEK.CMD.LOGS.**) UACC(READ) -
  DATA('z/OS Explorer - LOGS OPERATOR COMMAND')
RDEFINE FACILITY (FEK.CMD.LOGS.AUDIT.**) UACC(NONE) –
  DATA('z/OS Explorer - LOGS OPERATOR COMMAND')
PERMIT FEK.CMD.LOGS.AUDIT.** CLASS(FACILITY) -
  ID(SYSPROG) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Log collection – requirements for RSED started task

The **MODIFY LOGS** operator command uses the RSED started task user ID to collect host logs and setup information, and by default, user log files are created with secure file access permissions (only owner has access). To be able to collect secure user log files, the RSED started task user ID must be permitted to read them.

The OWNER argument of the **MODIFY LOGS** operator command results in the specified user ID becoming the owner of the collected data. In order to change ownership, the RSED started task user ID must be permitted to use the CHOWN z/OS UNIX service.

There are three ways these permissions can be provided to the RSED started task user ID. In order of preference, these are

- Access to select profiles in the UNIXPRIV class. This method is used in the FEKRACF sample job.
- Access to the BPX.SUPERUSER profile in the FACILITY class
- UID 0

UNIXPRIV class permits

The UNIXPRIV class holds profiles that allow a security administrator to selectively hand out special z/OS UNIX related permits, instead of granting all z/OS UNIX related permits with the super user approach.

Table 11. UNIXPRIV z/OS UNIX related permits

Profile	Permit	Result
SUPERUSER.FILESYS	READ	User is allowed to read any file or directory.

Table 11. UNIXPRIV z/OS UNIX related permits (continued)

Profile	Permit	Result
SUPERUSER.FILESYS.ACLOVERRIDE	READ	Permit is only required if ACLOVERRIDE is already defined. It allows the user to read any file or directory, regardless of ACL definitions.
SUPERUSER.FILESYS.CHOWN	READ	User is allowed to change the owner of any file or directory.

Note: When the SUPERUSER.FILESYS.ACLOVERRIDE profile is defined, access permissions defined in ACL (access Control List) take precedence over the permissions granted through SUPERUSER.FILESYS. The RSED started task user ID will need READ access permit to the SUPERUSER.FILESYS.ACLOVERRIDE profile to bypass ACL definitions.

BPX.SUPERUSER profile permit

When the RSED started task user ID has READ permission to the BPX.SUPERUSER profile in the FACILITY class, it is able to temporarily make itself a z/OS UNIX super user, for whom z/OS UNIX file access permissions do not count.

UID 0

When the RSED started task user ID has UID 0 specified in its OMVS segment, it is a z/OS UNIX super user, for whom z/OS UNIX file access permissions do not count. However, this approach is not advised since UID 0 is likely a shared UID, and it is advised to give the RSED started task user ID a unique UID due to other permissions granted to the ID. (For example, z/OS UNIX administrators require UID 0 for certain system management tasks.)

Miscellaneous information

GATE trashing

The first time an address space instructs RACF to accesses a resource class that is not RACLISTed (stored in memory), like the DATASET class, RACF will retrieve and store all the related generic profiles in the user's address space, in a list known as GATE (Generic Anchor Table Entry). Up to z/OS 1.12, RACF maintains four generic anchors for each address space and four for each MVS TCB that has its own ACEE. When all four are used up, RACF replaces the least recently referenced one when a new one comes in.

If your users frequently access more than four data set high-level qualifiers, the RSE thread pools (serving multiple users using threads with user-specific ACEEs) might experience GATE trashing as RACF has to rotate new entries through the available anchor slots.

In z/OS 1.12, RACF introduced the **GENERICANCHOR** option of the **SET** command, which allows you to increase the size of the table. This can be set system-wide or for each job name.

Managed ACEE

z/OS Explorer 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 z/OS Explorer) 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 z/OS Explorer.

ACEE caching

RACF can save ACEEs (Accessor Environment Elements) using VLF (Virtual Lookaside Facility) and retrieve them for later use. z/OS Explorer asks your security software to build multiple security environments (ACEEs) for the same user (one for each user-specific thread in the RSE thread pool), and can thus benefit from ACEE caching.

For more information on ACEE caching, see "ACEEs and VLF considerations" in the *Security Server RACF System Programmer's Guide* (SA22-7681).

TCP/IP port reservation

If you use the `EZB.PORTACCESS.<sysname>.<tcpname>.<resname>` security profile in the `SERVAUTH` class to control bind permissions to a given port, it is the client's user ID instead of the RSED daemon server ID that does the bind to the ports defined in `_RSE_PORTRANGE`. Consult your TCP/IP administrator to learn whether the ports in `_RSE_PORTRANGE` are protected and defined with the `SAF` keyword or not.

z/OS Explorer configuration files

There are several z/OS Explorer 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 18.
- `LIMIT_CONSOLE={LIMITED | NOLIMIT}`

  Define authority level of the EMCS console used for executing actions. For more information, see "Actions against jobs - console" on page 21.
- `LIMIT_VIEW={USERID | NOLIMIT}`

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

  Define whether JES Job Monitor can be accessed from outside this z/OS system. For more information, see section *FEJJCNFG, JES Job Monitor configuration file* in the *Basic customization* chapter of the *Host Configuration Guide* (SC27-8437).
- `APPLID={FEKAPPL | *}`

  Application ID used for PassTicket creation/validation. For more information, see "Using PassTickets" on page 15.
- `CONSOLE_NAME={JMON | &USERID | ...}`

Define which console name will be used when you issue JES operator commands. For more information, see “Actions against jobs - console” on page 21.

- `GEN_CONSOLE_NAME={ON | OFF}`

  Enables or disables automatic generation of alternative console names. For more information, see “Actions against jobs - console” on page 21.

Note: Details on these and other `FEJJCNFG` directives are available in "FEJJCNFG, JES Job Monitor configuration file" in the *Host Configuration Guide* (SC27-8437).

RSE - rse.env

- `_RSE_FEK_SAF_CLASS={FACILITY | *}`

  Security class holding `FEK.**` profiles. For more information, see “Push-to-client developer groups” on page 28 and “Altering client functions” on page 27.
- `(_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* (SC27-8437).
- `(_RSE_JAVAOPTS) –DHIDE_ZOS_UNIX={true | false}`

  Deny users access to z/OS UNIX. For more information, see "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC27-8437).
- `_(_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* (SC27-8437).
- `(_RSE_JAVAOPTS) -DAPPLID={FEKAPPL | *}`

  Application ID used for PassTicket creation/validation. For more information, see “Using PassTickets” on page 15.
- `(_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.
- ```
  GSK_PROTOCOL_SSLV3={ON | OFF}
  GSK_PROTOCOL_TLSV1={ON | OFF}
  GSK_PROTOCOL_TLSV1_1={ON | OFF}
  GSK_PROTOCOL_TLSV1_2={ON | OFF}
  ```

  Protocol selection for encrypted communication. For more information, see “Managing encryption protocols” on page 154.
- `GSK_V3_CIPHER_SPECS=*`
  ```

Cipher selection for encrypted communication. For more information, see “Managing encryption ciphers” on page 154.

- `GSK_FIPS_STATE={ON | OFF}`

  Use FIPS 140-2 compliant encrypted communication. For more information, see “(Optional) Enable FIPS 140-2 compliancy” on page 152.
- `(_RSE_JAVAOPTS) –Dlog.file.mode={RW.N.N | *}`

  File access permission mask of the host log files and directories.
- `(_RSE_JAVAOPTS) –Dlog.secure.mode={true | false}`

  Additional security checks (like ownership) for host log files and directories.
- `(_RSE_JAVAOPTS) -Ddaemon.log={/var/zexpl/logs | *}`

  Path leading to the audit log files. For more information, see “Audit logging” on page 17.
- `(_RSE_JAVAOPTS) -Daudit.log.mode={RW.R.N | *}`

  File access permission mask of the audit log files. For more information, see “Audit logging” on page 17.
- `(_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 17.

Note: Details on these and other `rse.env` directives are available in "rse.env, RSE configuration file" in the *Host Configuration Guide* (SC27-8437).

RSE - ssl.properties

- `daemon_keydb_file= SAF key ring name`

  Location of the RSE daemon certificate. For more information, see “Encrypted communication” on page 22.
- `daemon_key_label=certificate label`

  Name of the RSE daemon certificate. For more information, see “Encrypted communication” on page 22.
- `server_keystore_file=SAF key ring name`

  Location of the RSE server certificate. For more information, see “Encrypted communication” on page 22.
- `server_keystore_label=certificate label`

  Name of the RSE server certificate. For more information, see “Encrypted communication” on page 22.
- `server_keystore_type={JCERACFKS | JCECCARACFKS}`

  Type of key store used (Java key store or SAF key ring). For more information, see “Encrypted communication” on page 22.

Note: Details on these and other `ssl.properties` directives are available in "(Optional) ssl.properties, RSE encrypted communications" in the *Host Configuration Guide* (SC27-8437).

RSE - pushtoclient.properties

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

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

  Host-based control of z/OS Explorer client configuration files. For more information, see Chapter 7, “Push-to-client considerations,” on page 97.

- `product.enabled={true | false | SAF | LDAP}`
  `reject.product.updates={true | false | SAF | LDAP}`
  Host-based control of z/OS Explorer client product updates. For more information, see Chapter 7, “Push-to-client considerations,” on page 97.

Note: Details on these and other `pushtoclient.properties` directives are available in "(Optional) pushtoclient.properties, Host-based client control" in the *Host Configuration Guide* (SC27-8437).

Security definitions

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

`FEKRACF` is located in `FEK.#CUST.JCL`, unless you specified a different location when you customized and submitted the `FEK.SFEKSAMP(FEKSETUP)` job. See "Customization setup" in the *Host Configuration Guide* (SC27-8437) for more details.

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

Note: [1]

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

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

Requirements and checklist

To complete the security setup, the security administrator must know the values that are listed in Table 12. These values were defined during previous steps of the installation and customization of z/OS Explorer.

Table 12. Security setup variables

Description	• **Default value** • **Where to find the answer**	**Value**
z/OS Explorer product high-level qualifier	• `FEK` • SMP/E installation	

1. z/OS Explorer security definitions are based on IDz's.

Table 12. Security setup variables (continued)

Description	• **Default value** • **Where to find the answer**	Value
z/OS Explorer customization high-level qualifier	• `FEK.#CUST` • `FEK.SFEKSAMP(FEKSETUP)`, as described in "Customization setup" in the *Host Configuration Guide* (SC27-8437).	
JES Job Monitor started task name	• `JMON` • `FEK.#CUST.PROCLIB(JMON)`, as described in "PROCLIB changes" in the *Host Configuration Guide* (SC27-8437).	
RSE daemon started task name	• `RSED` • `FEK.#CUST.PROCLIB(RSED)`, as described in "PROCLIB changes" in the *Host Configuration Guide* (SC27-8437).	
Application ID	• `FEKAPPL` • `/etc/zexpl/rse.env`, as described in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC27-8437).	

The following list is an overview of the actions that are required to complete the basic security setup of z/OS Explorer. As documented in the following sections, different methods can be used to fulfill these requirements, depending on the required security level. For information about the security setup of optional z/OS Explorer services, see the previous sections.

- "Activate the security settings and classes" on page 39
- "Define an OMVS segment for z/OS Explorer users" on page 40
- "Define the z/OS Explorer started tasks" on page 40
- "Define RSE as a secure z/OS UNIX server" on page 41
- "Define the MVS program controlled libraries for RSE" on page 41
- "Define the PassTicket support for RSE" on page 42
- "Define the application protection for RSE" on page 43
- "Define z/OS UNIX file access permission for RSE" on page 43
- "Define the JES command security" on page 44
- "Define the data set profiles" on page 45
- "Verify the security settings" on page 46

Activate the security settings and classes

z/OS Explorer uses a variety of security mechanisms to ensure a secure and controlled host system environment for the client. 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 z/OS UNIX file access permission for RSE
  - `o SETROPTS GENERIC(UNIXPRIV)`
  - `o SETROPTS CLASSACT(UNIXPRIV) RACLIST(UNIXPRIV)`
- 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 the security software. In this case, you must merge the existing `*` and the new `**` definitions. 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 program control 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 z/OS Explorer users

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

In the following sample RACF commands, replace 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))
  ```

Define the z/OS Explorer started tasks

The following sample RACF commands create the `JMON`, and `RSED` started tasks, with protected user IDs (`STCJMON`, and `STCRSE`) and the `STCGROUP` group assigned to them.

- ```
  ADDGROUP STCGROUP OMVS(AUTOGID)
  DATA('GROUP WITH OMVS SEGMENT FOR STARTED TASKS')
  ```
- ```
  ADDUSER STCJMON DFLTGRP(STCGROUP) NOPASSWORD NAME('JES JOBMONITOR')
  OMVS(AUTOUID HOME(/tmp) PROGRAM(/bin/sh) )
  DATA('IBM Explorer for z/OS')
  ```
- ```
  ADDUSER STCRSE DFLTGRP(STCGROUP) NOPASSWORD NAME('RSE DAEMON')
  OMVS(AUTOUID HOME(/tmp) PROGRAM(/bin/sh) ASSIZEMAX(2147483647) )
  DATA('IBM Explorer for z/OS')
  ```
- ```
  RDEFINE STARTED JMON.* DATA('JES JOBMONITOR')
  STDATA(USER(STCJMON) GROUP(STCGROUP) TRUSTED(NO))
  ```
- ```
  RDEFINE STARTED RSED.* DATA('RSE DAEMON')
  STDATA(USER(STCRSE) 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 daemon 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. Set this value in the `ASSIZEMAX` variable of the OMVS segment for user ID `STCRSE`. Setting this value ensures that RSE daemon gets 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`. Setting the limit ensures that RSE gets the required thread limit, regardless of changes to `MAXTHREADS` or `MAXTHREADTASKS` in `SYS1.PARMLIB(BPXPRMxx)`. To determine the correct value for the thread limit, see "Tuning considerations" in the *Host Configuration Reference Guide* (SC27-8438).
- User ID `STCJMON` is another good candidate for setting `THREADSMAX` in the OMVS segment, because JES Job Monitor uses a thread per client connection.

Consider making the `STCRSE` user ID restricted. Users with the `RESTRICTED` attribute cannot access protected (MVS) resources that 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, define the RESTRICTED.FILESYS.ACCESS profile in the UNIXPRIV class with UACC(NONE). For more information about restricting user IDs, see *Security Server RACF Security Administrator's Guide* (SA22-7683).

Attention: If you use restricted user IDs, explicitly add the permission to access a resource by using the TSO **PERMIT** or the z/OS UNIX **setfacl** commands. The resources include those resources where the z/OS Explorer 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 the access before activating it on a production system.

Define RSE as a secure z/OS UNIX server

RSE requires UPDATE access to the BPX.SERVER profile to create or delete the security environment for the client's thread. Note that using UID(0) to bypass this requirement is not supported. This step is required for clients to be able to connect.

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

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

Define the MVS program controlled libraries for RSE

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

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

- RALTER PROGRAM ** UACC(READ) ADDMEM('SYS1.LINKLIB'//NOPADCHK)
- RALTER PROGRAM ** UACC(READ) ADDMEM('SYS1.CSSLIB'//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. The profile obscures and complicates the search path used by your security software. In this case, you must merge the existing * and the new ** definitions. Use 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 z/OS Explorer interacts with.

- System load library, for encrypted communication
  - SYS1.SIEALNKE

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 ISPF Gateway
  - ISP.SISPLPA
- REXX runtime library
  - REXX.*.SEAGLPA
- z/OS Explorer
  - FEK.SFEKLPA

Define the PassTicket support for RSE

The client's password or other means of identification, such as an X.509 certificate is used only to verify the identity upon connection. Afterward, PassTickets are used to maintain thread security. This step is required for clients to be able to connect.

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

```
• RDEFINE PTKTDATA FEKAPPL UACC(NONE) SSIGNON(KEYMASKED(key16))
  APPLDATA('NO REPLAY PROTECTION - DO NOT CHANGE')
  DATA('IBM Explorer for z/OS')
• RDEFINE PTKTDATA IRRPTAUTH.FEKAPPL.* UACC(NONE)
  DATA('IBM Explorer for z/OS')
• PERMIT IRRPTAUTH.FEKAPPL.* CLASS(PTKTDATA) ACCESS(UPDATE) ID(STCRSE)
• SETROPTS RACLIST(PTKTDATA) REFRESH
```

RSE supports using an application ID other than FEKAPPL. Uncomment and customize the "APPLID=FEKAPPL" option in rse.env to activate this, as documented in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC27-8437). 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 above. 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 might also be on the access list of the profile. 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. For JES Job Monitor, the application ID is set in the FEJJCNFG configuration file with the APPLID directive.
- If the system has a cryptographic product installed and available, you can encrypt the secured signon application key for added protection. To do so, use the KEYENCRYPTED keyword instead of KEYMASKED. For more information, see *Security Server RACF Security Administrator's Guide* (SA22-7683).

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

Define z/OS UNIX file access permission for RSE

The **MODIFY LOGS** operator command uses the RSED started task user ID to collect host logs and setup information. And by default, user log files are created with secure file access permissions (only owner has access). To be able to collect secure user log files, the RSED started task user ID must be permitted to read them.

The OWNER argument of the **MODIFY LOGS** operator command results in the specified user ID becoming the owner of the collected data. In order to change ownership, the RSED started task user ID must be permitted to use the CHOWN z/OS UNIX service.

- `RDEFINE UNIXPRIV SUPERUSER.FILESYS UACC(NONE) DATA('OVERRIDE UNIX FILE ACCESS RESTRICTIONS')`
- `RDEFINE UNIXPRIV SUPERUSER.FILESYS.CHOWN UACC(NONE) DATA('OVERRIDE UNIX CHANGE OWNER RESTRICTIONS')`
- `PERMIT SUPERUSER.FILESYS CLASS(UNIXPRIV) ACCESS(READ) ID(STCRSE)`
- `PERMIT SUPERUSER.FILESYS.CHOWN CLASS(UNIXPRIV) ACCESS(READ) ID(STCRSE)`
- `SETROPTS RACLIST(UNIXPRIV) REFRESH`

Note that when the SUPERUSER.FILESYS.ACLOVERRIDE profile is defined, access permissions defined in ACL (access Control List) take precedence over the permissions granted through SUPERUSER.FILESYS. The RSED started task user ID will need READ access permit to the SUPERUSER.FILESYS.ACLOVERRIDE profile to bypass ACL definitions.

Define the 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('IBM Explorer for z/OS')`
- `SETROPTS RACLIST(APPL) REFRESH`

Note:

- As described in more detail in "Define the PassTicket support for RSE" on page 42, RSE supports the using of an application ID other than FEKAPPL. The APPL class definition must match the actual application ID used by RSE.
- The client connection request succeeds if the application ID is not defined in the APPL class.
- The client connection request will fail only if the application ID is defined and the user lacks READ access to the profile.

Define the 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 *Host Configuration Guide* (SC27-8437).

The following sample RACF commands give z/OS Explorer users conditional access to a limited set of JES commands, which are Hold, Release, Cancel, and Purge. Users have only 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('IBM Explorer for z/OS')`
- `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 the security before activating it on a production system.

Table 13 and Table 14 show the operator commands issued for JES2 and JES3, and the discrete security profiles that can be used to protect them.

Table 13. 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 14. 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

Table 14. JES3 Job Monitor operator commands (continued)

Action	Command	OPERCMDS profile	Required access
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 be executed only 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. For more information, see "Actions against jobs - target limitations" in the *Host Configuration Reference Guide* (SC27-8438).
- Users can browse any spool file, unless LIMIT_VIEW=USERID is defined in the JES Job Monitor configuration file. For more information, see "Access to spool files" in *Host Configuration Reference Guide* (SC27-8438).
- Even if users are not authorized for these operator commands, they 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; for example, 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 the data set profiles

READ access for users and ALTER for system programmers is sufficient for most z/OS Explorer 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('IBM Explorer for z/OS - HLQ STUB')
• ADDSD  'FEK.*.**' UACC(READ)
  DATA('IBM Explorer for z/OS')
• PERMIT 'FEK.*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
• SETROPTS GENERIC(DATASET) REFRESH
```

Note:

- Protect FEK.SFEKAUTH against updates because this data set is APF-authorized. The same is true for FEK.SFEKLPA, but here because this data sets is program controlled.
- The sample commands in this publication and in the FEKRACF job assume that Enhanced Generic Naming (EGN) is active. When EGN is active, the ** qualifier can be used to represent any number of qualifiers in the DATASET class. Substitute ** with * if EGN is not active on your system. For more information about EGN, see *Security Server RACF Security Administrator's Guide* (SA22-7683).

Verify the 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`
- Started tasks
  - `LISTGRP  STCGROUP OMVS`
  - `LISTUSER STCJMON  OMVS`
  - `LISTUSER STCRSE   OMVS`
  - `RLIST STARTED JMON.*  ALL STDATA`
  - `RLIST STARTED RSED.*  ALL STDATA`
- RSE as a secure z/OS UNIX server
  - `RLIST FACILITY BPX.SERVER ALL`
- MVS program controlled libraries for RSE
  - `RLIST PROGRAM ** ALL`
- PassTicket support for RSE
  - `RLIST PTKTDATA FEKAPPL ALL SSIGNON`
  - `RLIST PTKTDATA IRRPTAUTH.FEKAPPL.* ALL`
- Application protection for RSE
  - `RLIST APPL FEKAPPL ALL`
- z/OS UNIX file access permission for RSE
  - `RLIST   UNIXPRIV SUPERUSER.FILESYS ALL`
  - `RLIST   UNIXPRIV SUPERUSER.FILESYS.CHOWN ALL`
- JES command security
  - `RLIST CONSOLE JMON ALL`
  - `RLIST OPERCMDS MVS.MCSOPER.JMON ALL`
  - `RLIST OPERCMDS JES%.** ALL`
- Data set profiles
  - `LISTGRP FEK`
  - `LISTDSD PREFIX(FEK) ALL`

Optionally, profiles can exist that direct the z/OS Explorer behavior for a specific user. These profiles match the `FEK.**` filter and are by default located in the `FACILITY` class. See the `_RSE_FEK_SAF_CLASS` directive in `rse.env`. You can use the **SEARCH** command to list the profile names. Use the **RLIST** command to show the details for a profile.

- `SEARCH CLASS(FACILITY) FILTER(FEK.**)`
- `RLIST FACILITY #profile-name ALL`

Chapter 3. TCP/IP considerations

z/OS Explorer 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.

Note that most z/OS Explorer functions are z/OS UNIX based, and thus TCP/IP will use the z/OS UNIX search order to find its configuration files. See Chapter 13, "Setting up TCP/IP," on page 157 for more information.

The following topics are covered in this chapter:

TCP/IP ports

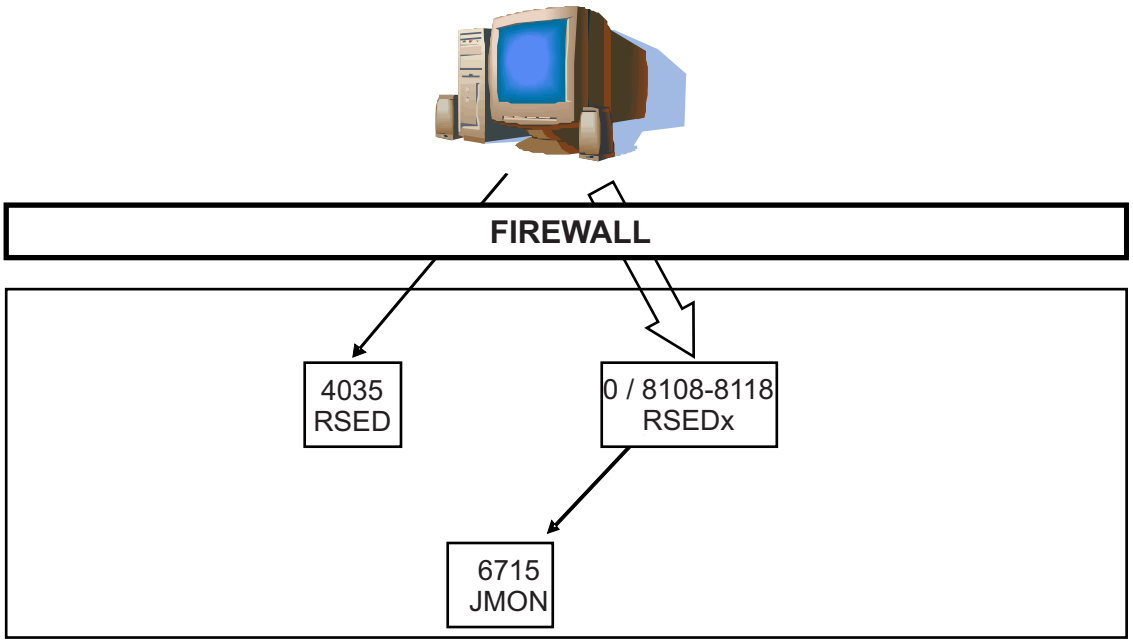


Figure 7. TCP/IP ports

Figure 7 shows the TCP/IP ports that can be used by z/OS Explorer. 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`. The port can be set in the `rse.env` configuration file. Communication on this port can be encrypted.

- RSE server for client-host communication. By default, any available port is used, but this can be limited to a specified range with the _RSE_PORTRANGE definition in rse.env. The default port range for _RSE_PORTRANGE is 8108-8118 (11 ports). Communication on this port can be encrypted.
- (optional) TN3270 Telnet service for the Host Connect Emulator, default port 23. Communication can be encrypted (default port 992). The default port assigned to the TN3270 Telnet service depends on whether or not the user chooses to use encryption.

Internal communication

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

- 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 rse.env configuration file.

TCP/IP port reservation

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

```
PORT      4035     TCP RSED   ; z/OS Explorer - RSE daemon
PORT      6715     TCP JMON   ; z/OS Explorer - JES job monitor
PORTRange 8108 11  TCP RSED*  ; z/OS Explorer - _RSE_PORTRANGE
```

If you use the SAF keyword in the PORT or PORTRANGE statement in PROFILE.TCPIP to control bind access, it is the client's user ID instead of the RSED daemon server ID that does the bind to the ports in _RSE_PORTRANGE.

LDAP considerations

RSE server can be configured to query one or more LDAP servers for various z/OS Explorer 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 rse.env.
- The LDAP server port numbers are listed in *_LDAP_PORT variables in rse.env.
- 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.

z/OS Explorer allows you to disable the delayed ACK function. On the host, this is done with the `DSTORE_TCP_NO_DELAY` directive in `rse.env`, as documented in the *Host Configuration Guide* (SC27-8437).

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 z/OS Explorer is not active on the default stack, then selected z/OS Explorer functions might fail. Using stack affinity is a sure way to resolve this. Stack affinity instructs z/OS Explorer 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 RSED started task by uncommenting and customizing the `_BPXK_SETIBMOPT_TRANSPORT` directive in the `rse.env` configuration file. See the related section in "Chapter 2 Basic Customization" of the *Host Configuration Guide* (SC27-8437) for more details on customizing this configuration file.

Distributed Dynamic VIPA

Distributed DVIPA (Dynamic Virtual IP Addressing) allows you to concurrently run identical z/OS Explorer 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 z/OS Explorer 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.
- When using a sysplex with an even number of systems, you should not use `ROUNDROBIN` as distribution method to do load balancing across your systems, unless you enable encrypted communication.

  z/OS Explorer clients first attempt to use encryption while setting up communication, and try again without encryption if this fails. This behavior for non-encrypted communication combined with `ROUNDROBIN` on a sysplex with an even number of systems will result in all uneven numbered systems (first, third, ...) being skipped, and all connections will go to the even numbered systems (second, fourth, ...).

- 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, z/OS Explorer 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 for the secondary connection. This implies that you cannot reserve ports for these connections in your TCP/IP profile with the `PORT` and `PORTRANGE` directives. You also cannot use `_RSE_PORTRANGE` in `rse.env` to limit the ports used by z/OS Explorer. z/OS Explorer does provide a workaround for this restriction, because this complicates firewall setup.

There are also some restrictions within z/OS Explorer when using distributed DVIPA:

- The `enable.dDVIPA` directive in `rse.env` must be enabled.
- To ensure that the z/OS Explorer client will not interfere with the correct port selection by TCP/IP, you should enable the `deny.nonzero.port` directive in `rse.env`.
- All participating z/OS Explorer servers must have an identical setup. You should share `/usr/lpp/IBM/zexpl` and `/etc/zexpl` among all participating systems. You should also share `/var/zexpl/pushtoclient`, if these directories are used. Note that `/var/zexpl/WORKAREA` and `/var/zexpl/logs` must be unique for each system.
- See Chapter 10, “Running multiple instances,” on page 123 to learn which z/OS Explorer components must be shared, and which ones must be unique per system.

JES Job Monitor and other z/OS Explorer 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. See *Communications Server: IP Configuration Reference* (SC31-8776) for more details on these directives.

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

Restricting port selection

The usage of `SYSPLEXPORTS` implies that TCP/IP will select an ephemeral port for the secondary connection. An ephemeral port is any port that is free and not

reserved in any way. The usage of an ephemeral port clashes with firewall best-practice to limit the ports that are opened for communication, because it is unknown which port will be used.

You can bypass this problem by forcing z/OS Explorer to use known ports for the secondary connection by defining a unique `_RSE_PORTRANGE` per system, and ensuring that the port ranges used are reserved for z/OS Explorer usage on all systems. You should note that this bypass requires TCP/IP APAR PM63379.

To ensure that TCP/IP will route the secondary connection to the correct system, z/OS Explorer must use a unique port range on each system. This implies that you cannot use a shared, identical, setup for the systems as `_RSE_PORTRANGE` in `rse.env` must be unique. See "Identical software level, different configuration files" on page 124 in Chapter 10, "Running multiple instances," on page 123 for information about how to set up multiple servers with different configuration files while using the same code. You should use a master copy of `rse.env` and a script to adjust and copy it to a system-specific setup to ensure the file remains identical across the different systems.

1. Set up z/OS Explorer on SYS1 as if it was a single system setup, but ensure that `/usr/lpp/IBM/zexpl` and `/etc/zexpl` are located in a shared file system. All MVS based parts should also be shared with SYS2.
2. Use `/etc/zexpl/rse.env` as the master copy and add a reference to `/etc/zexpl` to the end of the file so that the system-specific copies can pick up the remaining configuration files.

   ```
   $ oedit /etc/zexpl/rse.env
      -> add the following at the END:
        # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
        CFG_BASE=/etc/zexpl
        CLASSPATH=.:$CFG_BASE:$CLASSPATH
        # --
   ```

3. Create `/etc/zexpl/update.sh`, a shell script that will copy the master `rse.env` and adjust `_RSE_PORTRANGE`

   ```
   $ oedit /etc/zexpl/update.sh
   $ chmod 755 /etc/zexpl/update.sh
   ```

```
#! /bin/sh
# Licensed materials - Property of IBM
# 5655-EXP Copyright IBM Corp. 2012
# clone rse.env and set PORTRANGE for use with z/OS Explorer & DDVIPA

file=rse.env            #; echo file $file
sys=${1:-$(sysvar SYSNAME)}  #; echo sys $sys
dir=$(dirname $0)            #; echo dir $dir
# if sysname has a special char, precede it with \ (eg. SYS\$1)
case "$sys" in               # #### CUSTOMIZE THIS SECTION ####
  "SYS1") range=8108-8118;;
  "SYS2") range=8119-8129;;
esac                         #; echo range $range
echo "setting port range $range for $sys using $dir/$file"

if test ! $range ; then
  echo ERROR: no port range defined for $sys ; exit 12 ; fi
if test ! -e $dir/$file ; then
  echo ERROR: file $dir/$file does not exist ; exit 12 ; fi
if test ! -d $dir/$sys ; then
  echo ERROR: directory $dir/$sys does not exist ; exit 12 ; fi

mv $dir/$sys/$file $dir/$sys/prev.$file 2>/dev/null
sed="/_RSE_PORTRANGE/s/.*/_RSE_PORTRANGE=$range/"
sed "$sed" $dir/$file > $dir/$sys/$file

if test ! -s $dir/$sys/$file ; then
  echo ERROR creating $dir/$sys/$file, restoring backup
  mv $dir/$sys/prev.$file $dir/$sys/$file ; exit 8 ; fi
```

Figure 8. update.sh - support DDVIPA setup with a firewall

4. Create directories /etc/zexpl/SYS1 and /etc/zexpl/SYS2 and run /etc/zexpl/update.sh to populate the directories.

```
$ mkdir /etc/zexpl/SYS1 /etc/zexpl/SYS2
$ /etc/zexpl/update.sh SYS1
setting port range 8108-8118 for SYS1 using
   /etc/zexpl/rse.env
$ /etc/zexpl/update.sh SYS2
setting port range 8119-8129 for SYS2 using
   /etc/zexpl/rse.env
```

5. Ensure that the RSED started task points to /etc/zexpl/&SYSNAME.

```
//     CNFG='/etc/zexpl/&SYSNAME.'
```

Next, you must ensure that the defined port ranges are reserved for z/OS Explorer on all systems in the sysplex to ensure that the port number remains unique within the sysplex. Use the PORT or PORTRANGE statement in PROFILE.TCPIP to reserve all the ranges on every system. 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.

```
PORTRange 8108 22 RSED*          ; 8108-8129 - z/OS EXPLORER
                                 ; - secondary connection
```

As documented in "Connection flow" on page 5, the port range in _RSE_PORTRANGE can be small. 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.

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 z/OS Explorer distributed DVIPA.

After defining the distributed DVIPA, z/OS Explorer can be started on the systems to allow load balancing client connections across the systems. JES Job Monitor only interacts with the local RSE, and thus does not require a DVIPA setup. Clients will connect to port 4035 on IP address 10.10.10.1.

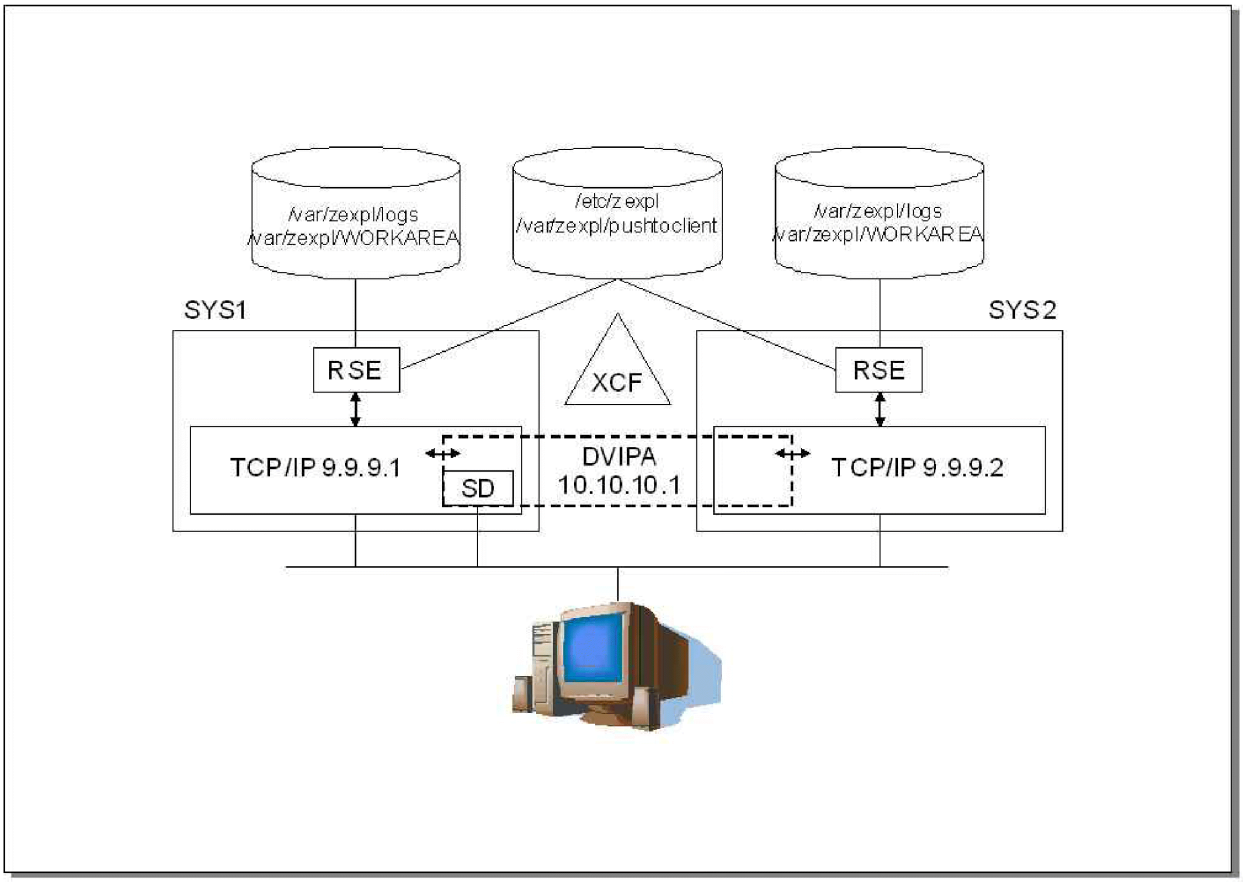


Figure 9. Distributed Dynamic VIPA sample

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 z/OS Explorer
  VIPADISTRIBUTE DEFINE
; VIPADISTRIBUTE makes 10.10.10.1 a distributed DVIPA, must match SYS2
    SYSPLEXPORTS                  ; z/OS Explorer prereq
    DISTMETHOD BASEWLM            ; BASEWLM
```

```
    10.10.10.1                    ; DVIPA address used by z/OS Explorer clients
    PORT 4035                     ; port used by z/OS Explorer clients
    DESTIP 9.9.9.1   9.9.9.2      ; z/OS Explorer 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 z/OS Explorer
  VIPADISTRIBUTE DEFINE
; VIPADISTRIBUTE makes 10.10.10.1 a distributed DVIPA, must match SYS1
    SYSPLEXPORTS                  ; z/OS Explorer prereq
    DISTMETHOD BASEWLM            ; BASEWLM
    10.10.10.1                    ; DVIPA address used by z/OS Explorer clients
    PORT 4035                     ; port used by z/OS Explorer clients
    DESTIP 9.9.9.1   9.9.9.2      ; z/OS Explorer active on SYS1 and SYS2
ENDVIPADYNAMIC
```

Chapter 4. WLM considerations

Unlike traditional z/OS applications, z/OS Explorer is not a monolithic application that can be identified easily to Workload Manager (WLM). z/OS Explorer consists of several components that interact to give the client access to the host services and data. As described in Chapter 1, "Understanding z/OS Explorer," 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

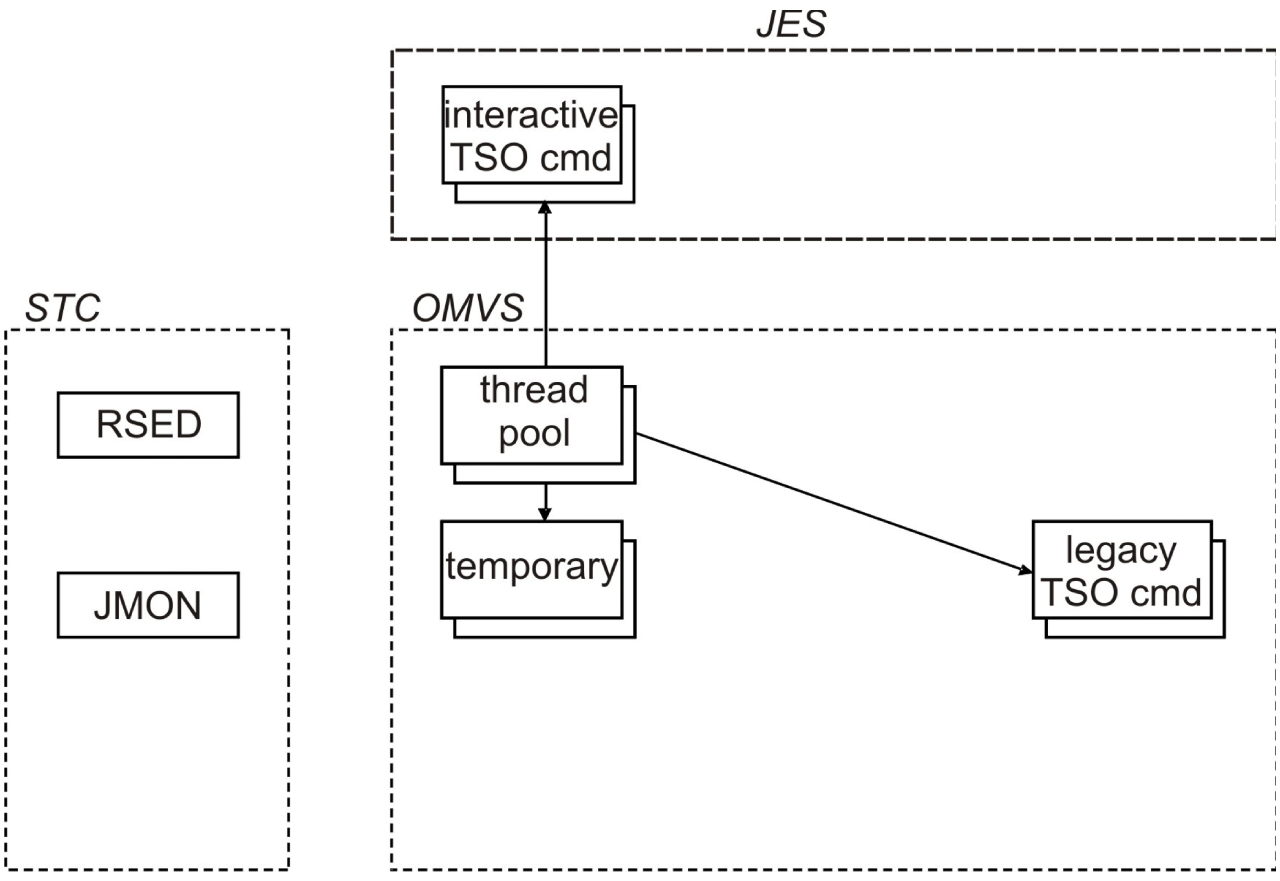


Figure 10. WLM classification

Figure 10 shows a basic overview of the subsystems through which z/OS Explorer workloads are presented to WLM.

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

As documented in "RSE as a Java application" on page 2, 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 z/OS Explorer, some workloads, such as the TSO Commands service (TSO cmd), 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.

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 15 lists the subsystem types that can receive z/OS Explorer workloads.

Table 15. 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.
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 16 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 16. WLM work qualifiers

		ASCH	JES	OMVS	STC
AI	Accounting Information	x	x	x	x
LU	LU Name (*)				
PF	Perform (*)		x		x
PRI	Priority		x		
SE	Scheduling Environment Name		x		
SSC	Subsystem Collection Name		x		
SI	Subsystem Instance (*)		x		
SPM	Subsystem Parameter				x

Table 16. WLM work qualifiers (continued)

		ASCH	JES	OMVS	STC
PX	Sysplex Name	x	x	x	x
SY	System Name (*)	x		x	x
TC	Transaction/Job Class (*)	x	x		
TN	Transaction/Job Name (*)	x	x	x	x
UI	User ID (*)	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, z/OS Explorer 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, z/OS Explorer 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 z/OS Explorer itself, not the whole system.

Table 17 lists the address spaces that are used by z/OS Explorer. z/OS UNIX will substitute "x" in the "Task Name" column by a random 1-digit number.

Table 17. WLM workloads

Description	Task name	Workload
JES Job Monitor	JMON	STC
RSE daemon	RSED	STC
RSE thread pool	RSEDx	OMVS
Interactive ISPF Gateway (TSO Commands serverice)	<userid>	JES
Legacy ISPF Gateway (TSO Commands service)	<userid>x	OMVS
TSO Commands service (APPC)	FEKFRSRV	ASCH
z/OS UNIX shell	<userid>	OMVS

Considerations for goal selection

The following general WLM considerations can help you to properly define the correct goal definitions for z/OS Explorer:

- 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 z/OS Explorer started tasks, RSE daemon and JES Job Monitor, are servicing real-time client requests.

Table 18. WLM workloads - STC

Description	Task name	Workload
JES Job Monitor	JMON	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.
- 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 19. WLM workloads - OMVS

Description	Task name	Workload
RSE thread pool	RSEDx	OMVS
Legacy ISPF Gateway (TSO Commands service)	<userid>x	OMVS
z/OS UNIX shell	<userid>	OMVS

- RSE thread pool

  An RSE thread pool is like the heart and brain of z/OS Explorer. Almost all data flows through here, and the miners (user specific threads) inside the thread pool control the actions of most other z/OS Explorer 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 Gateway, will report individual transactions to WLM, while others do not.

- Legacy ISPF Gateway

  The Legacy ISPF Gateway is an ISPF service invoked by z/OS Explorer to execute non-interactive TSO and ISPF commands explicitly issued by the client. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.
- z/OS UNIX shell

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

JES

JES-managed batch processes are used in various manners by z/OS Explorer. The most common usage is for TSO and ISPF commands via the Interactive ISPF Gateway.

Table 20. WLM workloads - JES

Description	Task name	Workload
Interactive ISPF Gateway	<userid>	JES

- Interactive ISPF Gateway

  The interactive ISPF Gateway is an ISPF service invoked by z/OS Explorer to execute interactive TSO and ISPF commands. Resource usage depends heavily on user actions, and will therefore fluctuate, but is expected to be minimal.

ASCH

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

Table 21. 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 z/OS Explorer to execute non-interactive TSO and ISPF commands. This includes explicit commands issued by the client as well as implicit commands issued by z/OS Explorer, 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.

Chapter 5. Tuning considerations

As explained in Chapter 1, "Understanding z/OS Explorer," on page 1, RSE (Remote Systems Explorer) is the core of z/OS Explorer. 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 z/OS Explorer setup. However, maintaining hundreds of users, each using multiple threads, a certain amount of storage, and possibly 1 or more address spaces requires proper configuration of both z/OS Explorer 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 z/OS Explorer, 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 z/OS Explorer itself. For example, resource usage of TN3270 is not documented.
- Adding third-party extensions to z/OS Explorer 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 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 z/OS Explorer. More details on the numbers presented here can be found in the next sections:

- "Address space count" on page 63
- "Process count" on page 64
- "Thread count" on page 67

Table 22 gives a general overview of the key resources used by the z/OS Explorer started tasks. These resources are allocated only once. They are shared among all z/OS Explorer clients.

Table 22. Common resource usage

Started task	Address spaces	Processes	Threads
JMON	1	1	4
RSED	1	3	17
RSEDx	(a) 2 + 2	2 + 3	3 + 15

Note: (a) There are two APF authorized address spaces and at least 1 RSE thread pool, which consists of two address spaces. Refer to "Address space count" on page 63 to determine the actual number of RSE thread pool address spaces.

Table 23 gives a general overview of the key resources used by requisite software. These resources are allocated for each z/OS Explorer client that invokes the related function.

Table 23. User-specific requisite resource usage

Requisite software	Address spaces	Processes	Threads
Interactive ISPF Gateway	1	1	2
Legacy ISPF Gateway	1	2	4
APPC	1	1	2

Table 24 gives a general overview of the key resources used by each z/OS Explorer client when executing the specified function. Non-numeric values, such as ISPF, are a reference to the corresponding value in Table 23.

Table 24. User-specific resource usage

User action	Address spaces	Processes	Threads		
	User ID	User ID	User ID	RSEDx	JMON
Logon	-	-	-	20	1
Timer for idle timeout	-	-	-	1	-
Search	-	-	-	1	-
Expand PDS(E)	ISPF	ISPF	ISPF	-	-
Open data set	ISPF	ISPF	ISPF	1	-

Table 24. 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	-

Note: ISPF can be substituted by APPC.

Address space count

Table 25 lists the address spaces that are used by z/OS Explorer, 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 25. Address space count

Count	Description	Task name	Shared	Ends after
1	JES Job Monitor	JMON	Yes	Never
1	RSE daemon	RSED	Yes	Never
2	RSE daemon APF authorized	RSEDx	Yes	Never
(a)	RSE thread pool	RSEDx	Yes	Never
(a)	RSE thread pool APF authorized	RSEDx	Yes	Never
1u	Interactive ISPF Gateway	<userid>	No	15 minutes or user logoff
lu	Legacy ISPF Gateway(TSO Commands service)	<userid>x	No	15 minutes or user logoff
lu	TSO Commands service (APPC)	FEKFRSRV	No	60 minutes or user logoff
(b)	Simultaneous Legacy ISPF Gateway usage by 1 user	<userid>x	No	Task completion
1u	z/OS UNIX shell	<userid>	No	User logoff

Note:

- (a) There is at least one RSE thread pool address space active. The actual number depends on:
  - The `minimum.threadpool.process` directive in `rse.env`. The default value is 1.
  - The number of users that can be serviced by one thread pool. The default settings aim for 10 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 `rse.env` is active.
- (b) z/OS Explorer has multiple threads active per user. In the event that the Legacy ISPF Gateway address space has not finished serving the request of one thread when another thread sends a request, ISPF will start up a new Gateway to process the new request. This address space ends after task completion.
- Most MVS data set related actions use the TSO Commands service, which can be active in the ISPF Gateway or an APPC transaction, respectively.

Use the formula in Figure 11 to estimate the maximum number of address spaces used by z/OS Explorer.

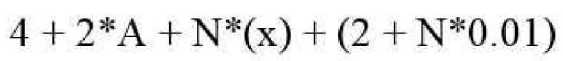


Figure 11. Maximum number of address spaces

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	TSO (Interactive ISPF Gateway)	TSO (Legacy ISPF Gateway)	TSO (APPC)
1	Yes	No	No
1	No	Yes	No
1	No	No	Yes

- "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 z/OS Explorer 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).

The definitions in Table 26 can limit the actual number of address spaces.

Table 26. Address space limits

Location	Limit	Affected resources
rse.env	maximum.threadpool.process	Limits the number of RSE thread pools
IEASYMxx	MAXUSER	Limits the number of address spaces
ASCHPMxx	MAX	Limits the number of APPC initiators for TSO Commands service (APPC)

Process count

Table 27 on page 65 lists the number of processes per address space that is used by z/OS Explorer. "u" In the "Address Spaces" column indicates that the amount must be multiplied by the number of concurrently active users using the function.

Table 27. Process count

Processes	Address spaces	Description	User ID
1	1	JES Job Monitor	STCJMON
3	1	RSE daemon	STCRSE
1+1	2	RSE daemon APF-authorized	STCRSE
2	(a)	RSE thread pool	STCRSE
1	(a)	RSE thread pool APF-authorized	STCRSE
1	1u	TSO (Interactive ISPF Gateway)	<userid>
2	(b)	TSO (Legacy ISPF Gateway)	<userid>
1	1u	TSO (APPC)	<userid>
1	1u	z/OS UNIX shell	<userid>

Note:

- (a) There is at least 1 RSE thread pool address space active. Refer to "Address space count" on page 63 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 Gateway active per user. The actual number can vary, as described in "Address space count" on page 63.
- Most MVS data set-related actions use the TSO Commands service, which can be active in the ISPF Gateway or an APPC transaction, respectively.
- 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 z/OS Explorer.

$$6 + 3*A + N*(x + z) + (10 + N*0.05)$$

Figure 13. Maximum number of processes

Where

- "6" 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	TSO (Interactive ISPF Gateway)	TSO (Legacy ISPF Gateway)	TSO (APPC)
1	Yes	No	No
2	No	Yes	No
1	No	No	Yes

- “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.
- "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 to estimate the maximum number of processes used by STCRSE, the RSED started task user ID (not counting the undocumented temporary processes).

$$6 + 3*A$$

Figure 14. Number of processes for STCRSE

Where

- "6" equals the number of processes used by the RSE daemon and RSE APF authorized address spaces.
- "A" represents the number of RSE thread pool address spaces.

Use the formula in Figure 15 to estimate the maximum number of processes used by a z/OS Explorer client (not counting the undocumented temporary processes).

$$(x + z)$$

Figure 15. 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 65).
- “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 65).

The definitions in Table 28 can limit the actual number of processes.

Table 28. 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
OMVS segment	PROCUSERMAX	Limits the number of processes for a user ID

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 `6 + 3*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.
- The PROCUSERMAX limit is unique per user ID, and is defined in your security software, in the OMVS segment of the user ID.

Thread count

Table 29 lists the number of threads used by selected z/OS Explorer 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 29. Thread count. This table lists the number of threads used by selected z/OS Explorer functions.

Threads			User ID	Description
RSEDx	**Active**	**Bootstrap**		
-	(f) 4 + 1u	-	STCJMON	JES Job Monitor
-	15	2	STCRSE	RSE daemon
-	-	-	STCRSE	RSE daemon APF authorized
-	2	-	STCRSE	RSE daemon APF authorized (send)
(a,g) 14 + 8u	-	(a) 1	STCRSE	RSE thread pool with single-threaded miners
(a,g) 14 + 19u	-	(a) 1	STCRSE	RSE thread pool, with multi-threaded miners
-	(a) 1	-	STCRSE	RSE thread pool APF authorized
-	2u	-	<userid>	TSO (Interactive Gateway)
-	(b) 4u	(b) 1u	<userid>	TSO (Legacy ISPF Gateway)
-	2u	-	<userid>	TSO (APPC)
6u	1u	-	STCRSE and <userid>	z/OS UNIX shell
(d) 1	-	-	STCRSE	Download
(e) 1	-	-	STCRSE	Search

Table 29. Thread count (continued). This table lists the number of threads used by selected z/OS Explorer functions.

Threads			User ID	Description
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 63 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 Gateway active per user. The actual number can vary, as described in "Address space count" on page 63.
- 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) Each download of host data will use a separate thread. This thread will end when the data is transferred to the client.
- (e) Each remote search will use a separate thread. This thread will end when the results are transferred to the client.
- 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.
- (f) A single user can have multiple active threads in JES Job Monitor to allow for concurrent processing of multiple requests.
- (g) User-specific miners can be started in two ways; all miners for a single user can share a single thread (dubbed single-threaded mode), or each miner uses a dedicated thread (dubbed multi-threaded mode). Grouping all miners for a user on a single thread reduces thread usage within the thread pool, but might cause some delays in command processing when a user is multi-tasking. The startup method is controlled by the `DSTORE_USE_THREADED_MINERS` directive in `rse.env`. The sample `rse.env` uses the multi-threaded mode.

Use the formula in Figure 16 to estimate the maximum number of threads used by an RSE thread pool in a single-threaded miner setup. Use the formula in Figure 17 to estimate the maximum number of threads used by an RSE thread pool in a multi-threaded miner setup. Use the formula in Figure 18 on page 69 to estimate the maximum number of threads used by JES Job Monitor.

$$14 + Np*(8 + x + z) + (20 + Np*0.1)$$

Figure 16. Maximum number of RSE thread pool threads (single-threaded miners)

$$14 + Np*(19 + x + z) + (20 + Np*0.1)$$

Figure 17. Maximum number of RSE thread pool threads (multi-threaded miners)

$$4 + N + (20 + N*0.1)$$

Figure 18. Maximum number of JES Job Monitor threads

Where

- "Np" represents the maximum number of concurrent users in this thread pool. The default settings aim for 10 users per thread pool.
- "N" represents the maximum number of concurrent users.
- "x" is one of the following values, depending on the selected configuration options.

Table 30. Values of x. This table lists the possible values of "x".

Option	Value
x	Idle timeout
0	No
1	Yes

- “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. Multiple concurrent downloads and searches are two examples that could require you to increase this buffer size.

The definitions in Table 31 can limit the actual number of threads in a process, which is mostly of importance for the RSE thread pools.

Table 31. Thread limits. This table lists the definitions that can limit the actual number of threads.

Location	Limit	Affected resources
OMVS segment	THREADSMAX	Limits the number of threads for a user ID
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.
rse.env	Xmx	Sets the maximum Java heap size. This storage is reserved and thus no longer available for thread-related control blocks.
rse.env	maximum.clients	Limits the number of clients (and thus their threads) in an RSE thread pool.
rse.env	maximum.threads	Limits the number of client threads in an RSE thread pool.
FEJJCNFG	MAX_THREADS	Limits the number of threads in JES Job Monitor.

Note:

- The THREADSMAX limit is unique per user ID, and is defined in your security software, in the OMVS segment of the user ID.

- The value for `maximum.threads` in `rse.env` must be lower than the value for `MAXTHREADS` and `MAXTHREADTASKS` in `BPXPRMxx`, and `THREADSMAX` in the OMVS segment of the RSED started task user ID.
- The **DISPLAY PROCESS,CPU** operator command, which shows the active threads in a thread pool, is limited to showing only the first 4000 threads.

Temporary resource usage

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

However, z/OS Explorer will temporarily use additional resources for housekeeping tasks and to satisfy the following requests:

- Processing an audit file event (`audit.action` directive in `rse.env`) uses one additional thread, one additional process, and possibly (if `audit.action.id` is set) one additional address space.
- Processing a logon event (`logon.action` directive in `rse.env`) uses one additional thread, one additional process, and possibly (if `logon.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 z/OS Explorer 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. Note that, to save CPU cycles, garbage collection tends to wait until the occupied storage is actually needed, thus leaving storage which is no longer used allocated (and becoming paged out) for a longer time than absolutely required.

The maximum Java heap size is defined in `rse.env` with the `Xmx` directive. You should specify a value of 256 MB or higher. When running in 64-bit mode, Java will attempt to allocate the heap above the 2 GB bar, freeing up space below the bar.

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 `rse.env` 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 140 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`
- `MEMLIMIT` in `SYS1.PARMLIB(SMFPRMxx)` for 64-bit addressing mode

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. When running in 64-bit mode, Java will attempt to allocate the heap above the 2GB bar, freeing up space below the bar.

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 z/OS Explorer. 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 z/OS Explorer 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.
- When running in 64-bit mode, ensure that the storage above the 2 GB bar is actually available to RSE.
- See "Address Space size" on page 140 to learn more about where address space size limits can be set.
- The default `rse.env` setup aims for 10 users per thread pool.
  - `maximum.clients=10`
  - `maximum.threads=250` (14+19*10 = 205), so 250 allows for 10 clients
- The default `rse.env` setup lets the Java heap grow up to 512 MB. This allows for 10 clients using an average of 51 MB per client (10*51 = 510).

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 `rse.env`. 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 `rse.env`. 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 z/OS Explorer 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=1,904MB

startup:
================================================
ProcessId(416     ) Memory Usage(17%) Clients(0)

Jobname      CPU Time     Storage       EXCP
-------      --------     -------     ------
JMON             0.01       10.4M        154
RSED             2.84       70.3M      67138
RSED5            0.72       71.7M      36552
================================================

logon 1:
================================================
ProcessId(175     ) Memory Usage(29%) Clients(1)

Jobname      CPU Time     Storage       EXCP
-------      --------     -------     ------
JMON             0.01       10.7M        229
RSED             2.98       70.5M      67451
RSED5            2.54      123.0M      59572
================================================

logon 2:
================================================
ProcessId(175     ) Memory Usage(35%) Clients(2)

Jobname      CPU Time     Storage       EXCP
-------      --------     -------     ------
JMON             0.02       10.8M        241
RSED             3.07       70.7M      67806
RSED5            3.31      127.0M      64720
================================================

logon 3:
================================================
ProcessId(175     ) Memory Usage(48%) Clients(3)

Jobname      CPU Time     Storage       EXCP
-------      --------     -------     ------
JMON             0.02       10.9M        249
RSED             3.11       70.7M      68047
RSED5            3.63      129.8M      68248
================================================

logon 4:
================================================
ProcessId(175     ) Memory Usage(44%) Clients(3)
ProcessId(16777515) Memory Usage(27%) Clients(1)

Jobname      CPU Time     Storage       EXCP
-------      --------     -------     ------
JMON             0.03       10.9M        268
RSED             3.19       70.6M      68465
RSED5            3.71      129.2M      68327
RSED8            3.10      124.0M      60370
================================================
```

Figure 19. Resource usage with 5 logons

```
logon 5:
================================================
ProcessId(175     ) Memory Usage(45%) Clients(3)
ProcessId(16777515) Memory Usage(36%) Clients(2)

Jobname      CPU Time     Storage       EXCP
-------      --------     -------     ------
JMON             0.04       11.0M        285
RSED             3.25       70.7M      68713
RSED5            3.74      129.2M      68357
RSED8            3.64      127.3M      64224
================================================
```

Figure 20. Resource usage with 5 logons (continued)

Figure 19 on page 73 and Figure 20 show a scenario where 5 clients log on to an RSE daemon with a 10 MB Java heap.

- A thread pool (RSED5) is in a dormant state upon startup, using about 71 MB, of which 1.7 MB is in the Java heap (17% of 10 MB).
- The thread pool becomes active when the first client connects, using another 50 MB plus 2 MB for each client that connects.
- Part of this 2 MB 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 fourth connection being created in a new thread pool (RSED8). 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 `rse.env`).

```
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
 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
 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
 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
 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 21. Resource usage while editing a PDS member

Figure 21 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 z/OS Explorer 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 129
- Auditing, as documented in "Audit logging" on page 17
- Push-to-client metadata, as documented in "Push-to-client metadata" on page 98.
- Temporary data

As documented in Chapter 11, "Troubleshooting configuration problems," on page 129, z/OS Explorer writes the RSE-related host logs to the following z/OS UNIX directories:

- `/var/zexpl/logs/server` for the RSE started task logs
- `/var/zexpl/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/zexpl/logs/server
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/zexpl/logs/server
total 144
-rw-------   1 STCRSE   STCGRP     36655 Jul 10 12:11 rsedaemon.log
-rw-------   1 STCRSE   STCGRP      1893 Jul 10 12:11 rseserver.log
$ ls -l /var/zexpl/logs/IBMUSER
total 160
-rw-------   1 IBMUSER  SYS1        3459 Jul 10 12:11 ffs.log
-rw-------   1 IBMUSER  SYS1           0 Jul 10 12:11 ffsget.log
-rw-------   1 IBMUSER  SYS1           0 Jul 10 12:11 ffsput.log
-rw-------   1 IBMUSER  SYS1         303 Jul 10 12:11 ffslock.log
-rw-------   1 IBMUSER  SYS1        7266 Jul 10 12:11 rsecomm.log

logoff
$ ls -l /var/zexpl/logs/server
total 80
-rw-------   1 STCRSE   STCGRP     36655 Jul 10 12:11 rsedaemon.log
-rw-------   1 STCRSE   STCGRP      2208 Jul 10 12:11 rseserver.log
$ ls -l /var/zexpl/logs/IBMUSER
total 296
-rw-------   1 IBMUSER  SYS1        6393 Jul 10 12:11 ffs.log
-rw-------   1 IBMUSER  SYS1           0 Jul 10 12:11 ffsget.log
-rw-------   1 IBMUSER  SYS1           0 Jul 10 12:11 ffsput.log
-rw-------   1 IBMUSER  SYS1         609 Jul 10 12:11 ffslock.log
-rw-------   1 IBMUSER  SYS1       45157 Jul 10 12:11 rsecomm.log

stop

$ ls -l /var/zexpl/logs/server
total 80
-rw-------   1 STCRSE   STCGRP     36655 Jul 10 12:11 rsedaemon.log
-rw-------   1 STCRSE   STCGRP      2490 Jul 10 12:12 rseserver.log
```

Figure 22. z/OS UNIX file system space usage

Figure 22 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. Audit logs are removed after the interval specified in `audit.retention.period` expires. The `keep.last.log` directive in `rse.env` changes this slightly, as it can instruct RSE to keep a copy of the previous logs. Older copies are always removed. If the `keep.all.logs` directive in `rse.env` is enabled, all logs have a timestamp appended to their name and the files are removed after the interval specified in `log.retention.period` expires.

A warning message is sent to the console when the file system holding the log files is running low on free space. This console message (FEK103E) is repeated regularly

until the low space issue is resolved. When the file system runs out of space, RSE will attempt to delete existing log files to free up space. Audit logs are not touched by this process.

The definitions in Table 32 control which data is written to the log directories, and where the directories are located.

Table 32. Log output directives

Location	Directive	Function
rescomm.properties	debug_level	Set the default log detail level.
rsecomm.properties	USER	Enable debug_level 2 for specified users.
rse.env	keep.all.logs	Keep a copy of the previous logs before startup/logon.
rse.env	keep.last.log	Keep a copy of the previous logs before startup/logon.
rse.env	enable.audit.log	Keep an audit trace of client actions.
rse.env	enable.standard.log	Write the stdout and stderr streams of the thread pool (or pools) to a log file.
rse.env	DSTORE_TRACING_ON	Enable DataStore action logging.
rse.env	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
Operator command	modify trace {on | off} USER=userid	Enable debug_level 2 for specified users.
Operator command	modify trace {on | off} SERVER=pid	Enable debug_level 2 for specified users.
Operator command	modify trace clear	Disable trace setup.
Operator command	modify logs	Collect host logs and setup information
rse.env	daemon.log	Home path for RSE started task and audit logs.
rse.env	user.log	Home path for user logs.
rse.env	CGI_ISPWORK	Home path for ISPF Gateway logs
rse.env	TMPDIR	Directory for IVP logs and **modify logs** operator command
rse.env	_CEE_DMPTARG	Directory for Java dumps

z/OS Explorer, together with requisite software such as the Legacy ISPF Gateway, also writes temporary data to `/tmp` and `/var/zexpl/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.

z/OS Explorer always tries to clean up these temporary files, but manual cleanup, as documented in "(Optional) WORKAREA and /tmp cleanup" in the *Host Configuration Guide* (SC27-8437), can be performed at virtually any time.

The definitions in Table 33 control where temporary data directories are located.

Table 33. Temporary output directives

Location	Directive	Function
rse.env	CGI_ISPWORK	Home path for temporary data.
rse.env	TMPDIR	Directory for temporary data.

Key resource definitions

/etc/zexpl/rse.env

The environment variables defined in `rse.env` are used by RSE, Java, and z/OS UNIX. The defaults used by z/OS Explorer are targeted at small to medium sized installations that do not require the optional components of z/OS Explorer. "rse.env, RSE configuration file" in the *Host Configuration Guide* (SC27-8437) describes each variable that is defined in the sample file, where the following variables require special attention:

`#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Xms128m -Xmx512m"`
Set initial (`Xms`) and maximum (`Xmx`) heap size. The defaults are `128M` and `512M` respectively. Uncomment and change to enforce the desired heap size values.

`#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dmaximum.clients=10"`
Maximum amount of clients serviced by one thread pool. The default is `10`. 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=250"`
Maximum amount of active threads in one thread pool to allow new clients. The default is `250`. 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 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 `rse.env` 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 z/OS Explorer:

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 multiple 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 16 MB line.
- Each client uses at least 17 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 z/OS Explorer 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 z/OS Explorer servers. This because the CPU time of all z/OS Explorer 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 z/OS Explorer is targeted at small to medium sized installations. "FEJJCNFG, JES Job Monitor configuration file" in the *Host Configuration Guide* (SC27-8437) 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 `7200`. 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 z/OS Explorer:

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 z/OS Explorer:

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 z/OS Explorer:

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 z/OS Explorer.

Monitoring

Since user workloads can change the need for system resources, the system should be monitored regularly to measure resource usage so that z/OS Explorer 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 z/OS Explorer, 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* (SC27-8437) 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         30
    MAXFILEPROC            83        103      64000
    MAXPROCUSER            97         99        200
    MAXTHREADS              9         14       1500
    MAXTHREADTASKS          9         14       1500
    REGION LIMITS:     CURRENT  HIGHWATER      LIMIT
    PRIVATE < 16M       72.0K          -       6.7M (6.9M)
    PRIVATE > 16M      610.8M          -    1731.0M (1811.0M)
    PRIVATE > 2G         2.0M       2.0M    NOLIMIT
  ```

  The CPU option of the **DISPLAY PROCESS** modify command will show the accumulated CPU usage (in milliseconds) of each thread in a thread pool:

  ```
  f rsed,appl=d p,cpu
  BPXM023I (STCRSE)
  ProcessId(33555087) ASId(002E) JobName(RSED8) Order(1)
  USERID   THREAD-ID        TCB@     ACC_TIME TAG
  STCRSE   0EDE540000000000 005E6B60      822 1/ThreadPoolProcess
  STCRSE   0EDE870000000001 005E69C8      001
  STCRSE   0EDE980000000002 005E6518     1814
  STCRSE   0EDEBA0000000003 005E66B0     2305
  STCRSE   0EDECB0000000004 005E62F8      001
  STCRSE   0EDEDC0000000005 005E60D8      001
  STCRSE   0EDF860000000006 005C2BF8      628 6/ThreadPoolMonitor$Memory
  UsageMonitor
  STCRSE   0EDF970000000007 005C2D90      003 7/ThreadPoolMonitor
  IBMUSER  0EE2C70000000024 005C08B0      050 38/JESMiner
  ```

- 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 z/OS Explorer 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 -Xms128m -Xmx512m -
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 -Xms128m -Xmx512m -
THREAD_ID        TCB@     PRI_JOB  USERNAME   ACC_TIME SC  STATE
0E08A000000000000 005E6DF0 OMVS                   .927 RCV  FU
0E08F00000000001 005E6C58                         .001 PTX JYNV
0E09300000000002 005E6AC0                        7.368 PTX JYNV
0E0CB00000000008 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 z/OS Explorer, 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 z/OS Explorer 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

z/OS Explorer 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/IBM/zexpl  (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
- using Legacy ISPF Gateway
- 3 hour inactivity time-out
- disallow usage of z/OS UNIX
- Foresee an average Java heap usage of 20 MB
- Users have unique z/OS UNIX UIDs
- Thread pools operate in multi-threaded miner mode

Thread pool count

By default, z/OS Explorer tries to add 10 users to a single thread pool using a default maximum Java heap of 512 MB, allowing an average of 51 MB heap usage per user (10*51 = 510). Our requirements indicate that we anticipate a lower average Java heap usage per user (20 MB), so we can increase the number of

clients per thread pool by setting `maximum.clients` to 20 (512/20 = 25). However, this also indicates that we should increase the `maximum.threads` from the default of 250 to 420 (14+20*(19+1)=414).

With 25 clients per thread pool and the need to support 500 connections, we now know we will need 20 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

  4 + 2*A + N*(x) + (2 + N*0.01)

  4 + 2*20 + 500*(1) + (2 + 500*0.01) = 551
- Address space count - per user

  x

  1
- Process count - maximum

  6 + 3*A + N*(x + z) + (10 + N*0.05)

  6 + 3*20 + 500*(2 + 0) + (10 + 500*0.05) = 1081
- Process count - STCRSE

  6 + 3*A

  6 + 3*20 = 66
- Process count - per user

  (x + z)

  (2 + 0) = 2
- Thread count - RSE thread pool

  14 + Np*(19 + x + z) + (20 + Np*0.1)

  14 + 25*(19 + 1 + 0) + (20 + 25*0.1) = 536
- Thread count - JES Job Monitor

  4 + N + (20 + N*0.1)

  4 + 500 + (20 + 500*0.1) = 574
- User IDs

  500 + 2 = 502

  The 2 extra user IDs are for STCJMON and STCRSE, the z/OS Explorer started task user IDs.

Defining limits

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

- /etc/zexpl/rse.env
  - Xmx512m

    not changed
  - Dmaximum.clients=25
  - Dmaximum.threads=420
  - Dminimum.threadpool.process=10

This change is optional; RSE will start new thread pools as needed
  - DDSTORE_USE_THREADED_MINERS=true
  - DHIDE_ZOS_UNIX=true
  - DDSTORE_IDLE_SHUTDOWN_TIMEOUT=10800000
- FEK.#CUST.PARMLIB(FEJJCNFG)
  - MAX_THREADS=574
- SYS1.PARMLIB(BPXPRMxx)
  - MAXPROCSYS(2000)

    1081 minimum, added extra buffer for tasks other than z/OS Explorer
  - MAXPROCUSER(100)

    66 minimum, added extra buffer in case the RSE thread pools support less than the projected 25 clients
  - MAXTHREADS(1500)

    must be minimum 574 for JES Job Monitor, which has the highest maximum thread count (STCRSE has 536)
  - MAXTHREADTASKS(1500)

    must be minimum 574 for JES Job Monitor, which has the highest maximum thread count (STCRSE has 536)
  - MAXUIDS(700)

    502 minimum, added extra buffer for tasks other than z/OS Explorer
  - MAXASSIZE(209715200)

    not changed (200 MB system default), we use ASSIZEMAX in the OMVS segment of user ID STCRSE
- SYS1.PARMLIB(IEASYSxx)
  - MAXUSER=2000

    551 minimum, added extra buffer for tasks other than z/OS Explorer
- 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 88, we can start monitoring the resource usage by z/OS Explorer to see if adjustment of some variables is needed. Figure 23 on page 90 shows the resource usage after 499 users logged on. (The example in the figure shows just the logging on. No user actions are indicated in the example.)

```
F RSED,APPL=D P
BPXM023I (STCRSE)
 ProcessId(83886168) Memory Usage(17%) Clients(25) Order(1)
 ProcessId(91      ) Memory Usage(17%) Clients(25) Order(2)
 ProcessId(122     ) Memory Usage(17%) Clients(25) Order(3)
 ProcessId(16777348) Memory Usage(17%) Clients(25) Order(4)
 ProcessId(16777358) Memory Usage(17%) Clients(25) Order(5)
 ProcessId(16777368) Memory Usage(17%) Clients(25) Order(6)
 ProcessId(16777378) Memory Usage(17%) Clients(25) Order(7)
 ProcessId(16777388) Memory Usage(17%) Clients(25) Order(8)
 ProcessId(16777398) Memory Usage(17%) Clients(25) Order(9)
 ProcessId(33554622) Memory Usage(17%) Clients(25) Order(10)
 ProcessId(16777416) Memory Usage(17%) Clients(25) Order(11)
 ProcessId(16777426) Memory Usage(17%) Clients(25) Order(12)
 ProcessId(16777436) Memory Usage(9%) Clients(25) Order(13)
 ProcessId(16777446) Memory Usage(17%) Clients(25) Order(14)
 ProcessId(16777456) Memory Usage(17%) Clients(25) Order(15)
 ProcessId(16777466) Memory Usage(17%) Clients(25) Order(16)
 ProcessId(16777476) Memory Usage(17%) Clients(25) Order(17)
 ProcessId(16777487) Memory Usage(17%) Clients(25) Order(18)
 ProcessId(16777497) Memory Usage(17%) Clients(25) Order(19)
 ProcessId(16777507) Memory Usage(16%) Clients(24) Order(20)

F RSED,APPL=D P,D
BPXM023I (STCRSE)
 ProcessId(83886168) ASId(0022) JobName(RSED857 ) Order(1)
  PROCESS LIMITS:    CURRENT  HIGHWATER      LIMIT
   JAVA HEAP USAGE(%)     17         17        100
   CLIENTS                25         25         25
   MAXFILEPROC           365        366      64000
   MAXPROCUSER            64         64        100
   MAXTHREADS            362        363       1500
   MAXTHREADTASKS        363        363       1500

TASID
 Jobname    Cpu time    Storage    EXCP
 --------  -----------  -------  ----------
 JMON             0.00     1780          73
 RSED             5.88    95.2M       41958
 RSED1            8.26   190.1M       58669
 RSED1            8.17   187.0M       58605
 RSED2            8.06   185.3M       58653
 RSED2            8.19   183.1M       60209
 RSED3            8.12   189.1M       58650
 RSED3            8.03   186.7M       58590
 RSED4            8.15   188.2M       58646
 RSED4            5.50   182.5M       58585
 RSED5            7.72   184.4M       58631
 RSED5            7.82   184.1M       58576
 RSED6            7.14   184.1M       58622
 RSED6            6.27   186.9M       58583
 RSED7            5.17   185.1M       58804
 RSED7            6.57   185.2M       58621
 RSED7            5.86   182.8M       58565
 RSED8            0.36     1560        2459
 RSED8            7.94   184.1M       58615
 RSED8            7.45   181.8M       58548
 RSED9            8.16   190.6M       58802
 RSED9            7.62   183.8M       58610
 RSED9            7.36   177.7M       57478
```

Figure 23. 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 z/OS Explorer.

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.

z/OS Explorer has a default of `STEPLIB=NONE` coded in `rse.env`, as described in "`rse.env`, RSE configuration file" in the Host Configuration Guide (SC27-8437). 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 rse.env, as described in rse.env, 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.

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 z/OS Explorer, Java command-line options are defined in the `_RSE_JAVAOPTS` directive of `rse.env`, as described in "Defining extra Java startup parameters with _RSE_JAVAOPTS" in the *Host Configuration Guide* (SC27-8437).

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

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 `rse.env`. 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" on page 94, 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

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

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

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

`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* (SC27-8437) for more information.

Typically, z/OS systems, developer workstations 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 z/OS Explorer client to create one or more client configurations, and by using IBM Installation Manager to create the response files used to update the z/OS Explorer client.

See the z/OS Explorer Knowledge Center for details about how the client administrator 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. z/OS Explorer client administrators are not able to export global configuration data unless the target system is a primary system. z/OS Explorer 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 z/OS Explorer configuration (`/etc/zexpl`) and push-to-client metadata (`/var/zexpl/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/zexpl/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 z/OS Explorer client components become eligible to be managed by push-to-client, more subdirectories are created dynamically. See the export wizard in the z/OS Explorer client (**File > Export... > IBM Explorer for z/OS> 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.

- `/var/zexpl/pushtoclient/install/` holds configuration files used to update the client product version upon connection to the host. The actual location is

specified in /var/zexpl/pushtoclient/keymapping.xml, which is created and maintained by a z/OS Explorer client administrator. The files within are maintained by a client administrator.

- /var/zexpl/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/zexpl/pushtoclient/keymapping.xml, which is created and maintained by a z/OS Explorer client administrator. The files within are maintained by a client administrator.

See "Customization setup" in the "Basic customization" chapter of the *Host Configuration Guide* (SC27-8437) 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 (PTCADMIN), assign it a unique GID (2), and make it the default group for user ID PTCADM1, which also receives a unique UID (6).

```
ADDGROUP PTCADMIN OWNER(IBMUSER) SUPGROUP(SYS1) –
  DATA('z/OS EXPLORER - CLIENT ADMIN')
ALTGROUP PTCADMIN OMVS(GID(2))
CONNECT PTCADM1 GROUP(PTCADMIN) AUTH(USE)
ALTUSER PTCADM1 DFLTGRP(PTCADMIN) OMVS(UID(6))
```

The following sample **chown** z/OS UNIX command changes the owner and group of /var/zexpl/pushtoclient and everything in it to PTCADM1 and PTCADMIN respectively. The command should be executed by a super-user (UID 0) to avoid permission problems.

```
chown -R ptcadm1:ptcadmin /var/zexpl/pushtoclient
```

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

```
chmod –R 775 /var/zexpl/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 9 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 z/OS Explorer client 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 z/OS Explorer client has a wizard, to be used by the client administrator, that can export the current configuration, which in turn is imported by all z/OS Explorer 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/zexpl/pushtoclient`).

Client version control

When a z/OS Explorer client 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.

Multiple developer groups

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 34.

*Table 34. Push-to-client group support matrix for *.enabled*

*.enabled value	Function enabled	Multiple groups supported
False	No	No
True	Yes	No

*Table 34. Push-to-client group support matrix for *.enabled (continued)*

*.enabled value	Function enabled	Multiple groups supported
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 35.

Table 35. 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 105 and "SAF-based group selection" on page 110 for details about the required setup for the respective function. See "(Optional) pushtoclient.properties, Host-based client control" in the *Host Configuration Guide* (SC27-8437) 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, z/OS Explorer limits which definitions will be used, based on a selection made by the user.

Table 36. Push-to-client group concatenations

Additional groups	Definitions used
None	Default
One	Default or (default + group)
Multiple	Default or (default + 1 group)

z/OS Explorer 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 configuration group (which can be the default group) and a specific product 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 configuration group binding connects to the host, and `config.enabled`indicates the push-to-client function is active, z/OS Explorer queries all configuration groups to determine to which groups the user belongs to and prompts the user to select a group. Upon successive connections, only the selected group is queried to see if the group membership is still valid.

Table 37. Workspace configuration group bindings

config.enabled	**Workspace bound to this configuration update group**
False	None
True	Default
LDAP	Default or Group (after prompting)
SAF	Default or Group (after prompting)

When a workspace that has no product group binding connects to the host, and `product.enabled` indicates the push-to-client function is active, z/OS Explorer queries all product groups to determine to which groups the user belongs to and prompts the user to select a group. Upon successive connections, only the selected group is queried to see if the group membership is still valid.

Table 38. Workspace product group bindings

product.enabled	**Workspace bound to this product update group**
False	None
True	Default
LDAP	Default or Group (after prompting)
SAF	Default or Group (after prompting)

The `reject.*.updates` directives can work with and without group definitions. If groups are used for reject.*.updates, then the group-binding of the related `*.enabled` directive is used. When an update is present, z/OS Explorer determines if the user is allowed to reject the update, and acts accordingly.

FEK.PTC.REJECT.*.UPDATES.sysname is used only for rejecting updates by workspaces bound to the default group. Workspaces bound to a group require you to be on the access list for FEK.PTC.REJECT.*.UPDATES.sysname.groupname to reject updates.

Group metadata location

As documented in "Metadata location" on page 98, all push-to-client metadata is stored in a directory structure on top of /var/zexpl/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/zexpl/pushtoclient/:

- Existing data in /var/zexpl/pushtoclient/ is interpreted as the data for the default group. Exporting to the default group creates or updates the metadata in /var/zexpl/pushtoclient/.
- Exporting to a group creates or updates the metadata in /var/zexpl/pushtoclient/grouping/<devgroup>, as if this were the base directory instead of /var/zexpl/pushtoclient/. The <devgroup> value matches the group name assigned to a specific group of developers.

Initial product customization creates the grouping/ directory in /var/zexpl/pushtoclient/. The client administrator is responsible for adding the <devgroup>/ directories to /var/zexpl/pushtoclient/grouping/.

Note that during initial product customization, the projects/, install/, and install/responsefiles/ directories are created in /var/zexpl/pushtoclient/. The client administrator must repeat these make-directory actions in /var/zexpl/pushtoclient/grouping/<devgroup>/ if there is a need for group-specific product upgrade scenarios.

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/zexpl/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.

Group name limitations

As documented in Group metadata location, group-specific metadata is stored in /var/zexpl/psuhtoclient/grouping/<devgroup>. The possible values that are assigned to <devgroup> are limited by the various components that make up the group support in push-to-client.

- Since the <devgroup> value must be defined in z/OS UNIX as a directory, you cannot use a forward slash (/) as part of your group-name.
- The z/OS Explorer client also creates a directory with the group name. Microsoft Windows does not allow the usage of the following characters in a directory

name; backslash (\), forward slash (/), colon (:), asterisk (*), question mark (?), double quotation mark ("), smaller than (<), larger than (>), and vertical bar (|).

- The method that is chosen to validate group membership also imposes restrictions on valid characters. For example, RACF does not allow the usage of characters with a special meaning to RACF; asterisk (*), percent (%), ampersand (&), blank ( ), comma (,), left round parenthesis (), right round parenthesis ()), and semicolon (;).

In the current implementation, z/OS Explorer also places some limitations on valid values:

- Even though lowercase characters are allowed in the `<devgroup>` directory name, z/OS Explorer converts everything to uppercase before validating the group membership.
- The only non-alphanumerical characters that are supported by z/OS Explorer for the `<devgroup>` name are the underscore (_), dash (-), and period (.). All other non-alphanumerical characters are not supported.

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.

   Note: For additional information on group names, see "Group name limitations" on page 103.
4. The client administrator creates the

   `/var/zexpl/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 z/OS Explorer 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. z/OS Explorer 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`), z/OS Explorer verifies membership of the group names listed in Table 39 to determine which developer groups the user belongs to, and whether a user is allowed to reject updates.

Table 39. 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 when the workspace is bound to the default group
FEK.PTC.REJECT.CONFIG.UPDATES.sysname.devgroup	User can reject configuration updates when the workspace is bound to the specified group
FEK.PTC.REJECT.PRODUCT.UPDATES.sysname	User can reject product updates when the workspace is bound to the default group

Table 39. Push-to-client LDAP information (continued)

Group name (cn=)	Result
FEK.PTC.REJECT.PRODUCT.UPDATES.sysname.devgroup	User can reject product updates when the workspace is bound to the specified group

The `devgroup` value matches the group name assigned to a specific group of developers. Note that the group name is visible on z/OS Explorer clients.

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

A user can select to bind a workspace to the default group for configuration updates if `config.enabled` in `pushtoclient.properties` is set to SAF or LDAP. If `config.enabled` is set to `TRUE`, the workspace is automatically bound to the default group.

A user can select to bind a workspace to the default group for product updates if `product.enabled` in `pushtoclient.properties` is set to `SAF` or `LDAP`. If `product.enabled` is set to TRUE, the workspace is automatically bound to the default group.

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 24 on page 107 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 24. Sample LDAP schema definition

LDAP server selection

There is a wide selection of commercial and free LDAP servers available. One example is the IBM Security 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" on page 106, 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 Security 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 z/OS Explorer LDAP objects just determine which choices a developer has in selecting a workspace layout and automatic z/OS Explorer client product upgrades.

LDAP server location

Any database server that supports the LDAP protocol can be used to host the z/OS Explorer push-to-client schema. Therefore, z/OS Explorer 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.

rse.env directive	**Default**
_RSE_LDAP_SERVER	Local host system

rse.env directive	Default
_RSE_LDAP_PORT	389
_RSE_LDAP_PTC_GROUP_SUFFIX	"O=PTC,C=zOSexplorer"

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 z/OS Explorer active on system CDFMVS08. IBM Security 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."

The following users use z/OS Explorer:

- Developers who work on banking applications, user ID BNK010 -> BNK014
- Developers who work on insurance applications, user ID INS010 -> INS014
- A z/OS Explorer client administrator, user ID PTCADM1

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 Security 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-z/OS EXPLORER
suffix "o=PTC,c=zOSexplorer"
databaseDirectory /var/ldap/ldbm/zexpl
```

2. Stop and start LDAP started task, GRDSRV, to pick up the configuration changes.
3. Create the /var/ldap/ldbm/zexpl directory.

   ```
   mkdir -p /var/ldap/ldbm/zexpl
   ```

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=zOSexplorer
   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 PTCADM1 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=zOSexplorer
objectclass: groupOfUniqueNames
cn: FEK.PTC.CONFIG.ENABLED.CDFMVS08.BANKING
description: z/OS EXPLORER push-to-client
# give client administrator access
uniqueMember: racfID=PTCADM1,profileType=user,cn=RACF,o=IBM,c=US

#  insurance workspace configuration
dn: cn=FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE,o=PTC,c=zOSexplorer
objectclass: groupOfUniqueNames
cn: FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE
description: z/OS EXPLORER push-to-client
# give client administrator access
uniqueMember: racfID=PTCADM1,profileType=user,cn=RACF,o=IBM,c=US

#  reject configuration updates
dn: cn=FEK.PTC.REJECT.CONFIG.UPDATES.CDFMVS08,o=PTC,c=zOSexplorer
objectclass: groupOfUniqueNames
cn: FEK.PTC.REJECT.CONFIG.UPDATES.CDFMVS08
description: z/OS EXPLORER push-to-client
# give client administrator access
uniqueMember: racfID=PTCADM1,profileType=user,cn=RACF,o=IBM,c=US

#  reject product updates
dn: cn=FEK.PTC.REJECT.PRODUCT.UPDATES.CDFMVS08,o=PTC,c=zOSexplorer
objectclass: groupOfUniqueNames
cn: FEK.PTC.REJECT.PRODUCT.UPDATES.CDFMVS08
description: z/OS EXPLORER push-to-client
# give client administrator access
uniqueMember: racfID=PTCADM1,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=zOSexplorer
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=zOSexplorer
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 PTCADM1 can reject configuration updates
reject.config.updates=LDAP
# only PTCADM1 can reject product updates
reject.product.updates=LDAP
```

rse.env

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=zOSexplorer"

/var/zexpl/pushtoclient/*install

While exporting the workspace configuration for groups BANKING and INSURANCE, the export wizard creates the /var/zexpl/pushtoclient/grouping/<devgroup>/ directories, and the directory structure behind it.

- /var/zexpl/pushtoclient/grouping/BANKING/*
- /var/zexpl/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/zexpl/pushtoclient/grouping/<devgroup>.

The client administrator must create the response files needed for product updates in the default-group directory, /var/zexpl/pushtoclient/install/responsefiles/.

SAF-based group selection

SAF (Security Access Facility) is an interface to access any z/OS security product. z/OS Explorer 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`), z/OS Explorer verifies access permits to the profiles listed in Table 40 to determine which developer groups the user belongs to, and whether a user is allowed to reject updates.

Table 40. 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 when the workspace is bound to the default group
FEK.PTC.REJECT.CONFIG. UPDATES.sysname.devgroup	30	READ	User can reject configuration updates when the workspace is bound to the specified group
FEK.PTC.REJECT.PRODUCT. UPDATES.sysname	31	READ	User can reject product updates when the workspace is bound to the default group
FEK.PTC.REJECT.PRODUCT. UPDATES.sysname.devgroup	31	READ	User can reject product updates when the workspace is bound to the specified group

Note: z/OS Explorer 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 z/OS Explorer clients.

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

A user can select to bind a workspace to the default group for configuration updates if `config.enabled` in `pushtoclient.properties` is set to `SAF` or `LDAP`. If `config.enabled` is set to `TRUE`, the workspace is automatically bound to the default group.

A user can select to bind a workspace to the default group for product updates if `product.enabled` in `pushtoclient.properties` is set to SAF or LDAP. If `product.enabled` is set to `TRUE`, the workspace is automatically bound to the default group.

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

By default, z/OS Explorer expects the `FEK.*` 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 z/OS Explorer to use this class instead. To do that, uncomment `_RSE_FEK_SAF_CLASS` in `rse.env` and provide the desired class name, for example XFACILIT.

Sample setup

Assume a company has z/OS Explorer 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
- PTCADM1 : z/OS Explorer 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

The profiles are defined in the `XFACILIT` class because the longest profile name, `FEK.PTC.REJECT.PRODUCT.UPDATES.CDFMVS08.DEVINSUR`, is 48 characters long, which is more than the 39 characters supported by the `FACILITY` class.

```
# allow PTCADMIN and DEVBANK to select push-to-client group BANKING
RDEFINE XFACILIT (FEK.PTC.CONFIG.ENABLED.CDFMVS08.BANKING) –
  UACC(NONE) DATA('z/OS EXPLORER - PUSH-TO-CLIENT')
PERMIT FEK.PTC.CONFIG.ENABLED.CDFMVS08.BANKING CLASS(XFACILIT) –
  ID(PTCADM1 DEVBANK) ACCESS(READ)

# allow PTCADMIN and DEVINSUR to select push-to-client group INSURANCE
RDEFINE XFACILIT (FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE) –
  UACC(NONE) DATA('z/OS EXPLORER - PUSH-TO-CLIENT')
PERMIT FEK.PTC.CONFIG.ENABLED.CDFMVS08.INSURANCE CLASS(XFACILIT) –
  ID(PTCADM1 DEVINSUR) ACCESS(READ)

# PTCADMIN can reject configuration updates on any system and for any group
RDEFINE XFACILIT (FEK.PTC.REJECT.CONFIG.UPDATES.**) –
  UACC(NONE) DATA('z/OS EXPLORER - PUSH-TO-CLIENT')
PERMIT FEK.PTC.REJECT.CONFIG.UPDATES.** CLASS(XFACILIT) –
  ID(PTCADM1) ACCESS(READ)

# PTCADMIN can reject product updates on any system system and for any group
RDEFINE XFACILIT (FEK.PTC.REJECT.PRODUCT.UPDATES.**) –
  UACC(NONE) DATA('z/OS EXPLORER - PUSH-TO-CLIENT')
```

```
PERMIT FEK.PTC.REJECT.PRODUCT.UPDATES.** CLASS(XFACILIT) –
  ID(PTCADM1) ACCESS(READ)

# activate changes
SETROPTS RACLIST(XFACILIT) REFRESH
```

pushtoclient.properties

```
# BANKING and INSURANCE have different configuration needs
config.enabled=SAF
# everyone receives product updates
product.enabled=TRUE
# only PTCADM1 can reject configuration updates
reject.config.updates=SAF
# only PTCADM1 can reject product updates
reject.product.updates=SAF
```

rse.env

```
 _RSE_FEK_SAF_CLASS=XFACILIT
```

/var/zexpl/pushtoclient/*install

While exporting the workspace configuration for groups BANKING and INSURANCE, the export wizard creates the `/var/zexpl/pushtoclient/grouping/<devgroup>/` directories, and the directory structure behind it.

- /var/zexpl/pushtoclient/grouping/BANKING/*
- /var/zexpl/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/zexpl/pushtoclient/grouping/<devgroup>/`.

The client administrator must create the response files needed for product updates in the default-group directory, `/var/zexpl/pushtoclient/install/responsefiles/`.

Grace period for rejecting changes

Assume that while the sample setup is active, a z/OS Explorer 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(XFACILIT) –
  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(XFACILIT) –
  ID(DEVBANK) DELETE

# activate changes
SETROPTS RACLIST(FACILITY) REFRESH
```

Note: The security administrator could also have created a `FEK.PTC.REJECT.PRODUCT.UPDATES.*.DEVBANK` profile with UACC(READ). This would permit all developers that bound their workspace to the DEVBANK group to reject product updates. The reject permit is not granted to developers who bound their workspace to the default group, even if they are a member of the DEVBANK group, as this is controlled by the `FEK.PTC.REJECT.PRODUCT.UPDATES.*` profile.

Chapter 8. User exit considerations

This chapter assists you with enhancing z/OS Explorer by writing exit routines.

z/OS Explorer provides exit points for select z/OS Explorer events. An exit point is a specific point in a function's processing where the function invokes an exit routine if one exists. You can write an exit routine to perform additional processing.

Note that unlike most traditional exit points, z/OS Explorer exit points do not allow you to change the behavior of the function. The exit routine, if one exists, is invoked asynchronously, after the function completes. z/OS Explorer processing does not wait for the exit routine to end, nor does it check the completion status.

User exit characteristics

User exit activation

User exits are activated with the `_RSE_JAVAOPTS <exit_point>.action` variables in `rse.env`, where `<exit_point>` represents a keyword identifying a specific exit point, as documented in "Available exit points" on page 117.

```
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -D<exit_point>.action=<user_exit>"
```

By default, all exit points are disabled. Uncomment and specify the full pathname of the user exit routine to enable the exit point.

```
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -D<exit_point>.action.id=<userid>"
```

By default, the user ID assigned to RSE daemon is used to execute the provided exit routine. Uncomment and specify a user ID to use the specified ID for executing the user exit. There is no need to specify a password, because RSE will generate a PassTicket to be used as password when it switches to the specified user ID.

Writing a user exit routine

User exit routines are invoked as a z/OS UNIX shell command with possibly one or more arguments. This implies that the exit routine you develop must be executable from the z/OS UNIX command line. Common coding techniques include z/OS UNIX shell script and z/OS UNIX REXX exec, but compiled code like C/C++ is also possible.

See *UNIX System Services User's Guide* (SA22-7801) to learn more about z/OS UNIX shell scripts. See *Using REXX and z/OS UNIX System Services* (SA22-7806) to learn more about z/OS UNIX-specific extensions to the REXX language.

The exit routine is likely to be executed by a user ID with special permits (such as the RSE started task user ID, which is allowed to generate PassTickets). It is therefore important that you limit update authority to the exit routine to avoid abuse. The following sample z/OS UNIX commands limits write authority to the owner only, while everyone can read and execute the script.

```
$ chmod 755 process_logon.sh
$ ls -l process_logon.sh
-rwxr-xr-x   1 IBMUSER  SYS1        2228 Feb 28 23:44 process_logon.sh
```

Definitions in `rse.env` are available to the user exit routine as environment variables.

RSE invokes the user exit routine with a single argument string. The argument string can be a single value or a single string which holds multiple blank delimited keywords and values. See "Available exit points" on page 117 for more details.

Console messages

z/OS Explorer uses console message ID `FEK910I` to display data related to user exits.

Invocation of the exit routine is marked with the following console message:

```
FEK910I <EXIT_POINT> EXIT: invoking <exit_point> processing exit
  in thread <thread_id>
```

All data written to `stdout` (**echo** command in a shell script, **say** command in a REXX exec) will be sent to the console:

```
FEK910I <EXIT_POINT> EXIT: <message>
```

Termination of the exit routine is marked with the following console message:

```
FEK910I <EXIT_POINT> EXIT: completed <exit_point> processing exit
 in thread <thread_id>
```

Executing with a variable user ID

z/OS Explorer allows you to run an exit routine with either the started task user ID or a specified user ID. However, you might want to execute some actions in the exit routine using another user ID, such as the client user ID in the logon exit routine. This can be accomplished using standard z/OS UNIX services, as shown in the following samples.

z/OS UNIX shell script

As documented in *UNIX System Services Command Reference* (SA22-7802), z/OS UNIX offers the **su** command to use the privileges of a superuser or another user. There are few things to keep in mind when using the **su** command.

- The user ID executing the **su** command must have READ permission to the `BPX.SRV.<userid>` profile in the `SURROGAT` class of your security product to be able to switch to the user ID identified by `<userid>` without specifying a password.
- The **su** command starts a new shell, so the remaining commands in your shell script will not be executed until the shell started by the **su** command exits. In order to stage commands to be executed in the new shell started by the **su** command, you can use the **echo** command to create the desired command and the pipe command character to feed it into the new shell, as shown in the following example. Note that standard shell scripting rules apply for escaping special characters.

```
#! /bin/sh
myID=ibmuser
echo a $(id)
echo 'echo b $(id)' | su -s $myID
echo "echo c \$(id)" | su -s $myID
cat /u/ibmuser/iefbr14
echo "submit /u/ibmuser/iefbr14" | su -s $myID
```

This sample logon exit, executed by the started task user ID, will result in the following console messages:

```
+FEK910I LOGON EXIT: invoking logon processing exit in thread 411
+FEK910I LOGON EXIT: a uid=8(STCRSE) gid=1(STCGROUP)
+FEK910I LOGON EXIT: b uid=1(IBMUSER) gid=0(SYS1)
+FEK910I LOGON EXIT: c uid=1(IBMUSER) gid=0(SYS1)
+FEK910I LOGON EXIT: //IEFBR14  JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
+FEK910I LOGON EXIT: //IEFBR14  EXEC PGM=IEFBR14
$HASP100 IEFBR14  ON INTRDR                          FROM STC03919
IBMUSER
IRR010I  USERID IBMUSER  IS ASSIGNED TO THIS JOB.
+FEK910I LOGON EXIT: JOB JOB03926 submitted from path '/u/ibmuser/iefbr14'
ICH70001I IBMUSER  LAST ACCESS AT 00:46:13 ON MONDAY, MARCH 19, 2012
$HASP373 IEFBR14  STARTED - INIT 2    - CLASS A - SYS CD08
IEF403I IEFBR14 - STARTED - TIME=00.46.14
+FEK910I LOGON EXIT: completed logon processing exit in thread 411
    IEFBR14     IEFBR14     IEFBR14     0000
IEF404I IEFBR14 - ENDED - TIME=00.46.14
$HASP395 IEFBR14  ENDED
$HASP309 INIT 2    INACTIVE ******** C=BA
```

z/OS UNIX REXX exec

As documented in *Using REXX and z/OS UNIX System Services* (SA22-7806), z/OS UNIX offers the **seteuid** SYSCALL command to set the effective UID of the current process. There are few things to keep in mind when using the **seteuid** command.

- The **seteuid** command uses the z/OS UNIX UID, not the MVS user ID. You must first determine the UID of the target user ID, which can be done with the **getpwnam** SYSCALL command.
- The user ID executing the **seteuid** command must have READ permission to the BPX.SRV.<userid> profile in the SURROGAT class of your security product to be able to switch to the user ID identified by <userid> without specifying a password. Note that when multiple user IDs share the same UID, there is no way to determine which one of the user IDs will be checked.

```
/* rexx */
myID='ibmuser'
say userid()
address SYSCALL 'getpwnam' myID 'pw.'
say pw.1 pw.2 pw.3 pw.4 pw.5
address SYSCALL 'seteuid' pw.2  /* PW_UID = 2 */
say retval errno errnojr
say userid()
```

This sample logon exit, executed by the started task user ID, will result in the following console messages:

```
+FEK910I LOGON EXIT: invoking logon processing exit in thread 515
+FEK910I LOGON EXIT: STCRSE
+FEK910I LOGON EXIT: IBMUSER  1 0 / /bin/sh
+FEK910I LOGON EXIT: 0 0 0
+FEK910I LOGON EXIT: IBMUSER
+FEK910I LOGON EXIT: completed logon processing exit in thread 515
```

Available exit points

The following exit points are provided by z/OS Explorer:

- "audit.action"
- "logon.action" on page 118

audit.action

- **Timing:**

The audit user exit is invoked when the active audit log file is closed. (Auditing continues as RSE switched to a new audit log file.)

- **Invocation arguments (1):**
  - <audit_log>: full path name of the audit log file that was closed
- **Sample:**

  /usr/lpp/zexpl/samples/process_audit.rex

  This sample z/OS UNIX REXX exec builds a batch job that will process the audit log that was closed.

logon.action

- **Timing:**

  The logon user exit is invoked when a user has completed the logon process.
- **Invocation arguments (6):**
  - -i <userid>: client user ID, case is as provided by the client
  - -u <user_log_path>: directory where this client's user logs are kept
  - -s <server_log_path>: directory where server logs are kept
  - -c <config_path>: directory where configuration files are kept
  - -b <binaries_path>: directory where z/OS Explorer is installed
  - -p <port>: RSE daemon port
- **Sample:**

  /usr/lpp/zexpl/samples/process_logon.sh

  This sample z/OS UNIX shell script writes a logon message to the console.

Chapter 9. Customizing the TSO environment

This chapter is provided to assist you with mimicking a TSO logon procedure by adding DD statements and data sets to the TSO environment in z/OS Explorer.

The TSO Commands service

The TSO Commands service is the z/OS Explorer 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 z/OS Explorer 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 z/OS Explorer, 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. When the Interactive ISPF Gateway is used to create the TSO Commands service, prompting for data is supported, but ISPF panels are not. A 3270 emulator, such as the Host Connect Emulator which is part of the z/OS Explorer client, is needed to execute these.

Access methods

z/OS Explorer provides a choice on how to access the TSO Commands service.

- ISPF Gateway service. This is the default method used in the provided samples.
- An APPC transaction. This method is deprecated.

Note:

- For historical reasons, z/OS Explorer supports using APPC to create the TSO Commands service. However, APPC usage by z/OS Explorer 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 z/OS Explorer library page.

Using the Legacy ISPF Gateway access method

ISPF.conf

The `ISPF.conf` configuration file (by default located in `/etc/zexpl/`) defines the TSO/ISPF environment used by z/OS Explorer. There is only one active `ISPF.conf` configuration file, which is used by all z/OS Explorer 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
isplLib=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 ISPF Gateway creates a temporary ISPF profile for the TSO Commands service. However, you can instruct the ISPF Gateway z to use a copy of an existing ISPF profile. The key here is the `_RSE_ISPF_OPTS` statement in `rse.env`.

```
#_RSE_ISPF_OPTS="$_RSE_ISPF_OPTS&ISPPROF=&SYSUID..ISPPROF"
```

Uncomment the statement (remove the leading number 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
- `&SYSNAME.` to substitute the system name as specified in the `IEASYMxx` parmlib member

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.

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 z/OS Explorer setups

If the allocation exec scenarios described in the previous sections cannot handle your specific needs, you can create different instances of z/OS Explorer's RSE communication server, with each instance using its own ISPF.conf file. The main drawback of the method described below is that z/OS Explorer 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/zexpl
$ mkdir /etc/zexpl/tso2
$ cp rse.env /etc/zexpl/tso2
$ cp ISPF.conf /etc/zexpl/tso2
$ ls /etc/zexpl/tso2
ISPF.conf          rse.env
$ oedit /etc/zexpl/tso2/rse.env
      -> change: _RSE_RSED_PORT=4037
      -> change: CGI_ISPCONF=/etc/zexpl/tso2
      -> change: -Ddaemon.log=/var/zexpl/logs/tso2
      -> change: -Duser.log=/var/zexpl/logs/tso2
      -> add at the END:
         # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES
         CFG_BASE=/etc/zexpl
         CLASSPATH=.:$CFG_BASE:$CLASSPATH
         # --
$ oedit /etc/zexpl/tso2/ISPF.conf
      -> change: change as needed
```

The commands in the previous example copy the z/OS Explorer configuration files that require changes to a newly created tso2 directory. The CGI_ISPCONF variable in rse.env 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 CGI_ISPWORK in rse.env) 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/zexpl/tso2 configuration files. Note that if _RSE_RSED_PORT is not changed in rse.env, the new started task must specify a new port as startup argument.

Refer to the *Host Configuration Guide* (SC27-8437) 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 z/OS Explorer 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 section to plan the coexistence of the different instances of z/OS Explorer, after which you can use this configuration guide to customize them.

Although it is possible to share certain parts of z/OS Explorer 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. z/OS Explorer 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/IBM/zexpl` are the high-level qualifier and path used during the installation of the product. `FEK.#CUST`, `/etc/zexpl` and `/var/zexpl` are the default locations used during the customization of the product (see "Customization setup" in the *Host Configuration Guide* (SC27-8437) for more information).
- You should install z/OS Explorer 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 z/OS Explorer 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 z/OS Explorer installation directory.
  - Archive: `cd /SYS1/usr/lpp/IBM/zexpl; tar -cSf /u/userid/zexpl.tar`
  - Restore: `cd /SYS2/usr/lpp/IBM/zexpl; tar -xSpf /u/userid/zexpl.tar`

Identical setup across a sysplex

z/OS Explorer configuration files (and code) can be shared across different systems in a sysplex, with each system running its own identical copy of z/OS Explorer, if a few guidelines are obeyed. Note that this information is for stand-alone z/OS Explorer 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 49.

- 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 `rse.env`, 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/zexpl/` and `/var/zexpl/pushtoclient/` can be shared across the sysplex, as z/OS Explorer uses them in read-only mode.

- Temporary data directories like /tmp/ and /var/zexpl/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/zexpl/pushtoclient.properties configuration file, you must also share the related metadata directory, /var/zexpl/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-encrypted access for on-site usage, and encrypted 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 z/OS Explorer installation, redo the customization steps for the parts that are different, using different data sets, directories, and ports to avoid overlapping the current setup.

There are two methods for setting up similar configurations using the same code. Which one to choose depends on the changes needed in /etc/zexpl/rse.env. If the only changes to rse.env are to provide a unique RSED port number and unique log locations, then you can use the method described in "Nearly identical rse.env." Use the method described in "Different rse.env" on page 125 if there are more changes to rse.env.

Nearly identical rse.env

For example, we have multiple CA Endevor® SCM instances. To access them, we must use multiple ssl.properties configuration files, as these specify the required allocations. An RSE daemon can use only one ssl.properties file, so we will need multiple RSE daemons to access the different CA Endevor® SCM instances. Since there are no functional rse.env updates, we can use this method to do the setup.

In this example, we can set up two RSE daemons that share the same rse.env file and use unique ssl.properties files. Other configuration files and other z/OS Explorer servers (for example JES Job Monitor) can be shared by both RSE daemons. This would result in the following actions:

```
$ cd /etc/zexpl
$ oedit rse.env
      -> # make port provided as started task argument known to the rest of this file
      -> _RSE_RSED_PORT=$RSE_PORT

      -> # create unique log directories for servers sharing this rse.env
      -> _RSE_JAVAOPTS="$_RSE_JAVAOPTS -Ddaemon.log=/var/zexpl/logs/$_RSE_RSED_PORT"
      -> _RSE_JAVAOPTS="$_RSE_JAVAOPTS -Duser.log=/var/zexpl/logs/$_RSE_RSED_PORT"

      -> add at the END:
         # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES IN CFG_BASE
         CFG_BASE=/etc/zexpl
         CLASSPATH=.:$CFG_BASE:$CLASSPATH
         # --
$ mkdir secure
$ ln –s ../rse.env secure/rse.env
$ cp ssl.properties secure/
```

```
$ oedit secure/ ssl.properties
      -> adjust as needed
$ ls –l secure/
total 16
-rwxr-xr-x   1 IBMUSER  SYS1        7037 Jun  9 10:12 ssl.properties
lrwxrwxrwx   1 IBMUSER  SYS1          15 Jun  9 10:11 rse.env -> ../rse.env
```

The commands in the preceding example create a generic `rse.env` and a unique symbolic link to it, together with unique `ssl.properties` files.

What is left now is creating two RSED started tasks that specify unique port numbers and use the newly created configuration directories. Keep in mind that started task names should be limited to 7 characters.

```
SYS1.PROCLIB(RSED)
//RSED     PROC IVP=,                          * 'IVP' to do an IVP test
//            PORT=4035,
//            CNFG='/etc/zexpl',
//            HOME='/usr/lpp/IBM/zexpl'
//*
//RSED     EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
// PARM='PGM &HOME./bin/rsed.sh &IVP -C&CNFG -P&PORT'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//         PEND
//*


SYS1.PROCLIB(RSEDSEC)
//RSED     PROC IVP=,                          * 'IVP' to do an IVP test
//            PORT=4036,
//            CNFG='/etc/zexpl/secure',
//            HOME='/usr/lpp/IBM/zexpl'
//*
//RSED     EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
// PARM='PGM &HOME./bin/rsed.sh &IVP -C&CNFG -P&PORT'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//         PEND
//*
```

Different `rse.env`

For example we have a local development team, and a remote team who must use encrypted communication and who are not allowed to save their host password on the client. The different encrypted communication settings can be configured using the previous method, but there is a functional change in `rse.env` (`DENY_PASSWORD_SAVE=true`), which requires us to use the following method.

In this example, 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 encrypted and non-encrypted instances. This would result in the following actions:

```
$ cd /etc/zexpl
$ mkdir /etc/zexpl/secure
$ cp rse.env /etc/zexpl/secure
$ cp ssl.properties /etc/zexpl/secure
$ ls /etc/zexpl/secure/
rse.env    ssl.properties
$ oedit /etc/zexpl/secure/rse.env
      -> change: _RSE_RSED_PORT=4047
      -> change: -Ddaemon.log=/var/zexpl/logs/secure
```

```
      -> change: -Duser.log=/var/zexpl/logs/secure
      -> uncomment: -DDENY_PASSWORD_SAVE=true
      -> add at the END:
         # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES IN CFG_BASE
         CFG_BASE=/etc/zexpl
         CLASSPATH=.:$CFG_BASE:$CLASSPATH
         # --
$ oedit /etc/zexpl/secure/ssl.properties
      -> change: change as needed
```

The commands in the preceding example copy the z/OS Explorer configuration files that require changes to a newly created `secure` directory. The `daemon.log` and `user.log` variables in `rse.env` 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 the new configuration directory. 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/zexpl/secure` configuration files.

Refer to the related sections in the *IBM Explorer for z/OS Host Configuration Guide* (SC27-8437) for more information on the actions shown previously in this section.

Note: When using this technique to create dependant clones, know that `ssl.properties` must always be cloned to the dependant directory, even if it doesn't change. `rse.env` Must also be copied, and at least the `_RSE_RSED_PORT` directive within must be changed.

Automated synchronizing

In "Different `rse.env`" on page 125, the `rse.env` changes between the non-encrypted and encrypted RSE daemon are minimal, which makes it possible to automate the process of keeping their `rse.env` files synchronized. This simplifies service roll-out, because only one `rse.env` file must be maintained. The automated synchronization described here uses some ideas documented in "Nearly identical `rse.env`" on page 124.

The following example dynamically determines the RSED port number, adds the RSED port number to the log directory names and updates the CLASSPATH so that clones will find the remaining configuration files. Then the example updates both started task to pass in the desired port number and enhances the started task JCL of the encrypted RSE daemon to clone the `rse.env` of the non-encrypted RSE daemon upon startup, updating the `DENY_PASSWORD_SAVE` variable in the process. Since the port number is embedded in the log directory name, it is automatically different between both daemons.

1. Prepare the master `rse.env`.

   ```
   $ oedit /etc/zexpl/rse.env
         -> change:_RSE_RSED_PORT=$RSE_PORT
         -> change: -Ddaemon.log=/var/zexpl/logs/$RSE_RSED_PORT
         -> change: -Duser.log=/var/zexpl/logs/$RSE_RSED_PORT
         -> add at the END:
            # -- NEEDED BY CLONES TO FIND THE REMAINING CONFIGURATION FILES IN_CFG_BASE
            CFG_BASE=/etc/zexpl
            CLASSPATH=.:$CFG_BASE:$CLASSPATH
   # --
   ```

2. Prepare the other configuration files (that are not `rse.env` files) that differ between the master (non-encrypted) and the clone (encrypted).

```
$ mkdir /etc/zexpl/secure
$ cp /etc/zexpl/ssl.properties /etc/zexpl/secure/
$ oedit /etc/zexpl/secure/ssl.properties
-> change: change as needed
```

3. Update existing RSED started task to pass in the port number.

```
SYS1.PROCLIB(RSED)
//RSED     PROC IVP=,                          * 'IVP' to do an IVP test
//            PORT=4035,
//            CNFG='/etc/zexpl',
//            HOME='/usr/lpp/IBM/zexpl'
//*
//RSED     EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
// PARM='PGM &HOME./bin/rsed.sh &IVP -C&CNFG -P&PORT'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//         PEND
//*
```

4. Create an RSED started task with a unique port number that will clone the base `rse.env` and alter the `DENY_PASSWORD_SAVE` variable. The sample shown here also has a sample instruction to change the value of a variable instead of uncommenting a line. When choosing a filter-key, keep in mind that it must be unique, and that JCL has a `100` character limitation for the PARM field.

```
SYS1.PROCLIB(RSEDSEC)
//*
//* RSE DAEMON &ndash; ENCRYPTED COMMUNICATION, DENY PASSWORD SAVE
//*
//RSED     PROC IVP=,                * 'IVP' to do an IVP test
//            PORT=4036,
//            HOME='/usr/lpp/IBM/zexpl',
//            CNFG='/etc/zexpl/secure'
//*
//UNCOMENT SET SED='"/DENY_PASSWORD/s!.*\(_RSE_JAVAOPTS=.*\)!\1!"'
//*CHANGE  SET SED='"/JAVA_HOME=/s/J6.0/J7.0/"'
//         SET FILE='rse.env'
//*
//* copy /etc/zexpl/rse.env to /etc/zexpl/secure/rse.env
//* and alter it
//*
//CLONE    EXEC PGM=BPXBATCH,REGION=0M,COND=(4,LT),
//  PARM='SH cd &CNFG;sed &SED ../&FILE>&FILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*
//* start RSED with the newly created rse.env
//*
//RSED     EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,COND=(4,LT),
//  PARM='PGM &HOME./bin/rsed.sh &IVP -C&CNFG -P&PORT'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//         PEND
//*
```

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 z/OS Explorer. 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 z/OS Explorer:

- 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, `rse.env`, `*.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 z/OS Explorer 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 z/OS Explorer 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/IBM/zexpl`. A valid sample would be `/service/usr/lpp/IBM/zexpl`.
  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/zexpl/rse.env` 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`. 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 `rse.env` and thus cannot be shared, since `rse.env` 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.

Chapter 11. Troubleshooting configuration problems

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

Messages and return codes generated by z/OS Explorer components are documented in the *IBM Explorer for z/OS Messages Guide* and the section *Troubleshooting* in z/OS Explorer KC.

In z/OS Explorer library page, you can also find the latest version of the z/OS Explorer documentation.

The z/OS Explorer Knowledge Center documents the z/OS Explorer client, and how it interacts with the host (from a client's perspective).

Valuable information can also be found in the Mainframe Dev.

Log and setup analysis using FEKLOGS

The RSED start task supports the **MODIFY LOGS** operator command to collect z/OS Explorer host logs and setup information. The collected data is placed in a z/OS UNIX file, `$TMPDIR/feklogs.%sysname.%jobname`, where `$TMPDIR` is the value of the `TMPDIR` directive in `rse.env` (default `/tmp`), `%sysname` is your z/OS system name and `%jobname` is the name of the RSED started task.

By default, only the server logs are collected. The following command options allow you to collect different logs.

Table 41. Command options. This table lists the command options that collect different logs.

Option	Description
USER	Collect log files for the specified user ID's.
AUDIT	Collect audit logs.
NOSERVER	Do not collect server logs.

By default, all available log files are collected. The **RANGE** command option allows you to limit this selection to those log files that were updated in the last given number of hours.

z/OS Explorer will query your security product for access permits to `FEK.CMD.LOGS.**` profiles to determine if the requestor is allowed to collect the

specified logs. By default, the requestor is the RSED started task user ID, unless the OWNER option is specified. Only the requestor has access to the file holding the collected data.

To collect data before the RSED started task can start, z/OS Explorer provides a sample job, FEKLOGS, which gathers all z/OS UNIX log files as well as z/OS Explorer 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* (SC27-8437) 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. Do note that the output data set must be allocated as VB 2051 (default value in SDSF is VB 240) to avoid record truncation.

Log files

z/OS Explorer 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 rse.env, and $LOGNAME is the logon user ID (uppercase). If the user.log directive is a null string, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID.
  - .dstoreMemLogging - DataStore memory usage logging
  - .dstoreTrace - DataStore action logging
  - .dstoreHashmap.* - snapshot of the active DataStore hashmap
  - .dstoreStackTrace.* - snapshot of the active DataStore threads and where they were invoked
  - 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
  - ffslock.log - The log of the lock manager, that locks/unlocks a sequential data set or a PDS member
  - 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)

  Note:

  - 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 z/OS Explorer client, select the **Window > Preferences... > Remote Systems > Files** preference page and enable “Show hidden files”.
- daemon-home/server/

  The RSE daemon and RSE thread pool specific log files are located in `daemon-home/server`, where `daemon-home` is the value of the `daemon.log` directive in `rse.env`.
  - `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 the directory referenced by `TMPDIR`, if this variable is defined in `rse.env`. If the variable is not defined, the files are created in `/tmp`. The **MODIFY LOGS** operator command for the RSED started task also creates its output in this directory.
  - `fekfivpi.log` - The log of the `fekfivpi` IVP test
  - `feklogs.*` - Output of the **MODIFY LOGS** operator command

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* (SC27-8437) 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 135 for more details.

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 `rse.env`.
  - `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 rse.env 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 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* (SC27-8437) for more information.
- The rse*.log files can also exist with a ".last" extension instead of a ".log" extension if keep.last.log=true is specified in rse.env. By default, the ".last" log files are not created.
- The rse*.log files will have an extended name if keep.all.logs=true is specified in rse.env. By default the extended name is used. The following is a sample extended name, where RSED represents the address space name of the RSE daemon and yyyymmddhhmmss is a date and time stamp (year, month, day, hour, minute, second): rseserver.RSED#yyyymmddhhmmss.log

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 rse.env, and $LOGNAME is the logon user ID (uppercase). If the user.log directive is a null string, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID.

  - .dstoreMemLogging - DataStore memory usage logging
  - .dstoreTrace - DataStore action logging
  - .dstoreHashmap.* - snapshot of the active DataStore hashmap
  - .dstoreStackTrace.* - snapshot of the active DataStore threads and where they were invoked
  - 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
  - ffslock.log - The log of the lock manager, that locks or unlocks a sequential data set or a PDS member
  - 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)

Note:

- 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 z/OS Explorer 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* (SC27-8437).
- The `.dstore*` log files are created in UTF8. Use z/OS UNIX command **iconv -f UTF8 -t IBM-1047 .dstore*** to display them in EBCDIC (when using code page IBM-1047).
- Unlike all `*.log` files, the `.dstore*` log files are not removed automatically upon client reconnect. Removing these files is a manual action.
- 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* (SC27-8437) for more information.
- The `ffs*.log` and `rsecomm.log` files can also exist with a “`.last`” extension instead of a “`.log`” extension if `keep.last.log=true` is specified in `rse.env`. By default, the “`.last`” log files are not created.
- The `ffs*.log` and `rsecomm.log` files will have an extended name if `keep.all.logs=true` is specified in `rse.env`. By default the extended name is used. The following is a sample extended name, where RSEDx represents the address space name of the thread pool in which the user is active and `yyyymmddhhmmss` is a date and time stamp (year, month, day, hour, minute, second): `ffs.RSEDx#yyyymmddhhmmss.log`

fekfivpi IVP test logging

- **/tmp/fekfivpi.log**

  Output of the `fekfivpi -file` command (ISPF Gateway related IVP test). The log will be created in the directory referenced by `TMPDIR`, if this variable is defined in `rse.env`. 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 z/OS Explorer 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 `rse.env`. 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.

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 42. 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 135.

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."

Note that z/OS Explorer provides an operator command to trigger this dump. See the "Operator commands" chapter in the *Host Configuration Guide* (SC27-8437) for more details.

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."

Note that z/OS Explorer provides an operator command to trigger this dump. See the "Operator commands" chapter in the *Host Configuration Guide* (SC27-8437) for more details.

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 rse.env 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* (SC27-8437).

- Starting the JMON started task with the PRM=-TV parameter activates verbose mode (tracing).
- The **modify trace** and **modify message** operator commands let you select the desired detail level for log messages.

RSE tracing

There are several log files created by the components related to RSE. Most are located in `userlog/$LOGNAME/`, where `userlog` is the combined value of the `user.log` and `DSTORE_LOG_DIRECTORY` directives in `rse.env`, and `$LOGNAME` is the logon user ID (uppercase). If the `user.log` directive is a null string, the home path of the user is used. The home path is defined in the OMVS security segment of the user ID.

The amount of data written to `ffs*.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* (SC27-8437) and "(Optional) RSE tracing" in the *Host Configuration Guide* (SC27-8437) 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* (SC27-8437).

Note:

- 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 client, select the **Window > Preferences... > Remote Systems > Files** preference page and enable “Show hidden files”.
- The `.dstore*` log files are created in UTF8. Use z/OS UNIX command **iconv -f UTF8IBM-1047 .dstore*** to display them in EBCDIC (when using code page IBM-1047).
- Unlike all `*.log` files, the `.dstore*` log files are not removed automatically upon client reconnect. Removing these files is a manual action.

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 `rse.env`.

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* (SC27-8437) and "(Optional) RSE tracing" in the *Host Configuration Guide* (SC27-8437) for more details.

`serverlogs.count`, `stderr.*.log`, and `stdout.*.log` are only created if the `enable.standard.log` directive in `rse.env` is active, or if the function is dynamically activated with the **modify rsestandardlog on** operator command..

z/OS UNIX permission bits

z/OS Explorer 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 z/OS Explorer 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 z/OS Explorer 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 z/OS Explorer from creating the user's security environment, and will fail the connection request. Other indicators for this setup problem are:

- console message "FEK999E The module, fekfomvs must be marked as APF-authorized"
- PassTicket IVP fails with "ICH409I 282-010 ABEND DURING RACHECK PROCESSING"

Similar errors (such as messages BPXP014I and BPXP015I) can be expected if the file systems hosting Java or z/OS UNIX binaries are mounted with the NOSETUID parameter.

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 z/OS Explorer 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 13. This permission bit might get lost if you did not preserve it during a manual copy of the z/OS Explorer directories.

The following z/OS Explorer files must be program controlled:

- /usr/lpp/IBM/zexpl/bin
  - fekfdivp
  - fekfomvs
  - fekfrivp
- /usr/lpp/IBM/zexpl/lib/
  - fekfdir.dll
  - fekfdir64.dll
  - libfekdcore.so
  - libfekdcore64.so
  - libfekfmain.so
  - libfekfmain64.so
- /usr/lpp/IBM/zexpl/lib/icuc/
  - libicudata.dll
  - libicudata50.1.dll
  - libicudata50.dll
  - libicudata64.50.1.dll
  - libicudata64.50.dll
  - libicudata64.dll
  - libicuuc.dll
  - libicuuc50.1.dll
  - libicuuc50.dll

- libicuuc64.50.1.dll
- libicuuc64.50.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/IBM/zexpl
$ 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/IBM/zexpl
$ 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 z/OS Explorer 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 z/OS Explorer directories.

The following z/OS Explorer files must be APF-authorized:

- /usr/lpp/IBM/zexpl/bin/
  - fekfldsl.rex
  - fekfomvs
  - fekfrivp
  - fekfrmsg
  - fekftso.rex
  - feklogs.rex
  - send

Note: The following load modules in the SFEKAUTH load library must also be APF authorized:

- FEJJMON
- FEKEESX0

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/IBM/zexpl
$ 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/IBM/zexpl
$ 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).

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 z/OS Explorer configuration.

IPv4

```
MVS TCP/IP NETSTAT CS VxRy       TCPIP Name: TCPIP           16:36:42
User Id  Conn     Local Socket          Foreign Socket         State
-------  ----     ------------          --------------         -----
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
-------  ----     -----
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 z/OS Explorer 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 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.

SYSPLEX

z/OS Explorer is not SYSPLEX aware, and therefore requires that user-specific components that are started as JCL and are active on the same system the z/OS Explorer client is connected to.

In z/OS 2.1, you can specify SYSAFF=* or SYSTEM=* on the JOB card to enforce that the job runs on the system it was submitted. On older systems you must explicitly specify the correct system name when using a JESPLEX to unite multiple JES subsystems in a SYSPLEX.

Miscellaneous information

System limits

SYS1.PARMLIB(BPXPRMxx) defines many z/OS UNIX related limitations, which might be reached when several z/OS Explorer 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 rse.env, 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 rse.env, 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 (encrypted communication) 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.

Chapter 12. Setting up encrypted communication and X.509 authentication

This section is provided to assist you with some common problems that you may encounter when setting up encrypted communication, or during checking or modifying an existing setup. This section 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. TLS (Transport Layer Security) 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.

The actions needed to set up encrypted communications for z/OS Explorer 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 section we will clone the current RSE definitions, so that we have a 2nd RSE daemon connection that will use encrypted communication. We will also create our own security certificates to be used by the different parts of the RSE connection.

Throughout this section, a uniform naming convention is used:

- Certificate : `rsecert`
- Key and certificate storage : `keyring.racf`
- 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 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 is implemented as 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 application and uses a key store file which is implemented as a key ring managed by your security software.

Note:

- 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 `rse.env`.

  ```
  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.
- 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.

Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for information on RACF and digital certificates. For more details about ICSF and how to control who can use cryptographic keys and services see the *Cryptographic Services ICSF Administrator's Guide (SA22-7521).*

Create a key ring with RACF

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.

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 z/OS Explorer client, if the client already trusts the CA.

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

```
#  permit RSE daemon to access certificates
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)

#  refresh to make the changes visible
SETROPTS RACLIST(FACILITY) REFRESH

# create key ring
RACDCERT ID(stcrse) ADDRING(keyring.racf)
#  create self-signed certificate
RACDCERT ID(stcrse) GENCERT SUBJECTSDN(CN('zexpl rse') +
  OU('zexpl') O('IBM') L('Raleigh') SP('NC') C('US')) +
  NOTAFTER(DATE(2027-05-21)) WITHLABEL('rsecert') KEYUSAGE(HANDSHAKE)

## (optional) additional steps required to use a signed certificate
## 1. create a signing request for the self-signed certificate
##    The signing request will be placed in 'dsn'.
##    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.
# RACDCERT ID(stcrse) GENREQ (LABEL('rsecert')) DSN(dsn)
## 2. send the signing request to your CA of choice
## 3. check if the CA credentials (also a certificate) are already known
# RACDCERT CERTAUTH LIST
## 4. mark the CA certificate used to sign your certificate as trusted
# RACDCERT CERTAUTH ALTER(LABEL('CA cert')) TRUST
##    or add the CA certificate used to sign yours to the database
# RACDCERT CERTAUTH ADD(dsn) WITHLABEL('CA cert') TRUST
## 5. add the signed certificate to the database;
##    this will replace the self-signed one
# RACDCERT ID(stcrse) ADD(dsn) WITHLABEL('rsecert') TRUST
## 6. add the CA certificate to the key ring
# RACDCERT ID(stcrse) CONNECT(CERTAUTH LABEL('CA cert') +
#  RING(keyring.racf))

RACDCERT ID(stcrse) CONNECT(LABEL('rsecert') RING(keyring.racf) +
  DEFAULT USAGE(PERSONAL))

#  refresh to make the changes visible
SETROPTS RACLIST(DIGTCERT) REFRESH
```

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 secure RSE daemon. The next step is creating a new, self-signed, certificate with label rsecert. This certificate is then added to a newly created key ring (keyring.racf). The steps required to use a signed certificate are also listed.

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.

The result can be verified with the following `list` and `listring` options:

```
RACDCERT ID(stcrse) LIST
Digital certificate information for user STCRSE:

 Label: zexpl rse
 Certificate ID: 2QjW1OXi0sXZ1aaEqZmihUBA
 Status: TRUST
 Start Date: 2007/05/24 00:00:00
 End Date:   2027/05/21 23:59:59
```

```
 Serial Number:
      >00<
 Issuer's Name:
      >CN=CA cert.OU=CA.O=IBM.L=Raleigh.SP=NC.C=US<
 Subject's Name:
      >CN=zexpl rse.OU=zexpl.O=IBM.L=Raleigh.SP=NC.C=US<
 Private Key Type: Non-ICSF
 Private Key Size: 1024
 Ring Associations:
   Ring Owner: STCRSE
   Ring:
      >keyring.racf<

RACDCERT ID(stcrse) LISTRING(keyring.racf)
Digital ring information for user STCRSE:

  Ring:
     >keyring.racf<
  Certificate Label Name              Cert Owner     USAGE      DEFAULT
  --------------------------------    ------------   --------   -------
  rsecert                             ID(STCRSE)     PERSONAL     YES
  CA cert                             CERTAUTH       CERTAUTH     NO
```

Clone the existing RSE setup

In this step a new instance of the RSE configuration files is created, so that the secure setup can run parallel with the existing one(s). The following sample commands expect the configuration files to be in `/etc/zexpl/`, which is the default location used in "Customization setup" in the *Host Configuration Guide* (SC27-8437). Note that `rse.env` is the starting point for any configuration setup, and multiple setups implies multiple `rse.env` files.

```
$ cd /etc/zexpl
$ mkdir secure
$ ln –s ../rse.env secure/rse.env
$ cp ssl.properties secure
$ ls secure
rse.env    ssl.properties
```

The z/OS UNIX commands listed in the preceding example create a subdirectory called `secure` 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 encryption-specific.

By reusing most of the existing configuration files, we can focus on the changes that are actually required for setting up encrypted communication and avoid doing the complete RSE setup again. (For example, we can avoid defining a new location for `ISPF.conf`.)

The duplication of `rse.env` is done through a symbolic link, which points to the `rse.env` of the base, non-encrypted, RSE setup. This because there are no functional changes required to `rse.env`, it only needs a unique port number for RSE daemon and unique log locations, all of which can be resolved dynamically with proper preparation. This also simplifies future maintenance, as now only one version of `rse.env` must be maintained.

Update rse.env to enable coexistence

As mentioned in the previous step, `rse.env` of the non-encrypted setup is also used by the encrypted setup through the symbolic link. So we must update the file to dynamically pick up the correct port number and use unique log directories. We must also update it so that all setups are able to find their local and the shared configuration files. All this can be addressed by minor changes to `rse.env`.

```
$ oedit /etc/zexpl/rse.env
      -> # make port provided as started task argument known to the rest of this file
      -> _RSE_RSED_PORT=$RSE_PORT

      -> # create unique log directories for servers sharing this rse.env
      -> _RSE_JAVAOPTS="$_RSE_JAVAOPTS -Ddaemon.log=/var/zexpl/logs/$_RSE_RSED_PORT"
      -> _RSE_JAVAOPTS="$_RSE_JAVAOPTS -Duser.log=/var/zexpl/logs/$_RSE_RSED_PORT"

      -> add at the END:
         # -- NEEDED TO FIND THE REMAINING CONFIGURATION FILES IN CFG_BASE
         CFG_BASE=/etc/zexpl
         CLASSPATH=.:$CFG_BASE:$CLASSPATH
```

The changes in the preceding example pick up the port number provided as startup-argument and 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 secure RSE processes will first search the current directory (`/etc/zexpl/secure`) for configuration files and then search the original directory (`/etc/zexpl`).

Update ssl.properties to enable encryption

By updating `ssl.properties`, RSE is instructed to start using encrypted communication.

```
$ oedit /etc/zexpl/secure/ssl.properties
      -> change: enable_ssl=true
      -> uncomment and change: daemon_keydb_file=keyring.racf
      -> uncomment and change: daemon_key_label=rsecert
      -> uncomment and change: server_keystore_file=keyring.racf
      -> uncomment and change: server_keystore_label=rsecert
      -> uncomment and change: server_keystore_type=JCERACFKS
```

The changes in the preceding example enable encryption and tell the RSE daemon and RSE server that their (shared) certificate is stored under label `rsecert` in key ring `keyring.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 z Systems cryptographic hardware, when available. To be able to share the daemon definitions with the server when using ICSF, 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.

Update the existing RSE daemon

We updated `rse.env` to use the port-number provided as a startup argument, so we must also update the started task to provide the port number.

```
//*
//* RSE DAEMON
//*
//RSED     PROC IVP=,                   * 'IVP' to do an IVP test
//            PORT=4035,
//            CNFG='/etc/zexpl',
//            HOME='/usr/lpp/IBM/zexpl'
//*
//RSED     EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
// PARM='PGM &HOME./bin/rsed.sh &IVP -C&CNFG -P&PORT'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//          PEND
//*
```

Activate encryption by creating a new RSE daemon

As stated before, we will create a second connection that will use encrypted communication, 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* (SC27-8437):

1. Create a new member `FEK.#CUST.PROCLIB(RSEDSEC)` and copy in sample JCL `FEK.#CUST.PROCLIB(RSED)`.
2. Customize `RSEDSEC` by adding a job card on top and an exec statement at the bottom. Also provide the new port number and the location of the encryption-related configuration files (`/etc/zexpl/secure`), 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.

```
//RSEDSEC  JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),USER=STCRSE
//*
//* RSE DAEMON – ENCRYPTED COMMUNICATION
//*
//RSED     PROC IVP=,                        * 'IVP' to do an IVP test
//            PORT=4047,
//            CNFG='/etc/zexpl/secure',
//            HOME='/usr/lpp/IBM/zexpl'
//*
//RSED     EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
// PARM='PGM &HOME./bin/rsed.sh &IVP -C&CNFG -P&PORT'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//          PEND
//*
//RSED     EXEC RSED
//*
```

Figure 25. RSEDSEC - RSE daemon user job for encrypted communication

Note: The user ID assigned to the `RSEDSEC` 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 host configuration is complete and the RSE daemon for encrypted communication can be started by submitting job `FEK.#CUST.PROCLIB(RSEDSEC)`, which was created earlier.

The new setup can now be tested by connecting with the z/OS Explorer client. Since we created a new configuration for use by encrypted communication (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 z/OS Explorer client does not recognize the host certificate or the CA that signed it, z/OS Explorer client will prompt the user asking if this certificate can be trusted.

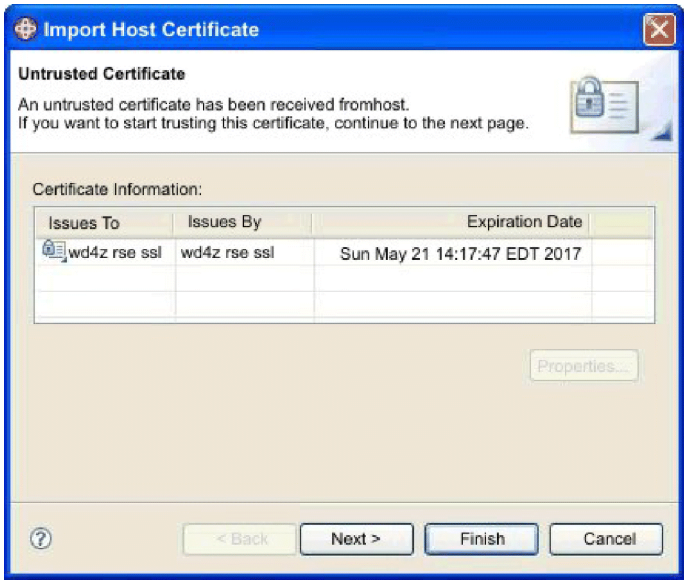


Figure 26. Import Host Certificate dialog

By clicking the Finish button the user can accept this certificate as trusted, after which the connection initialization continues.

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:

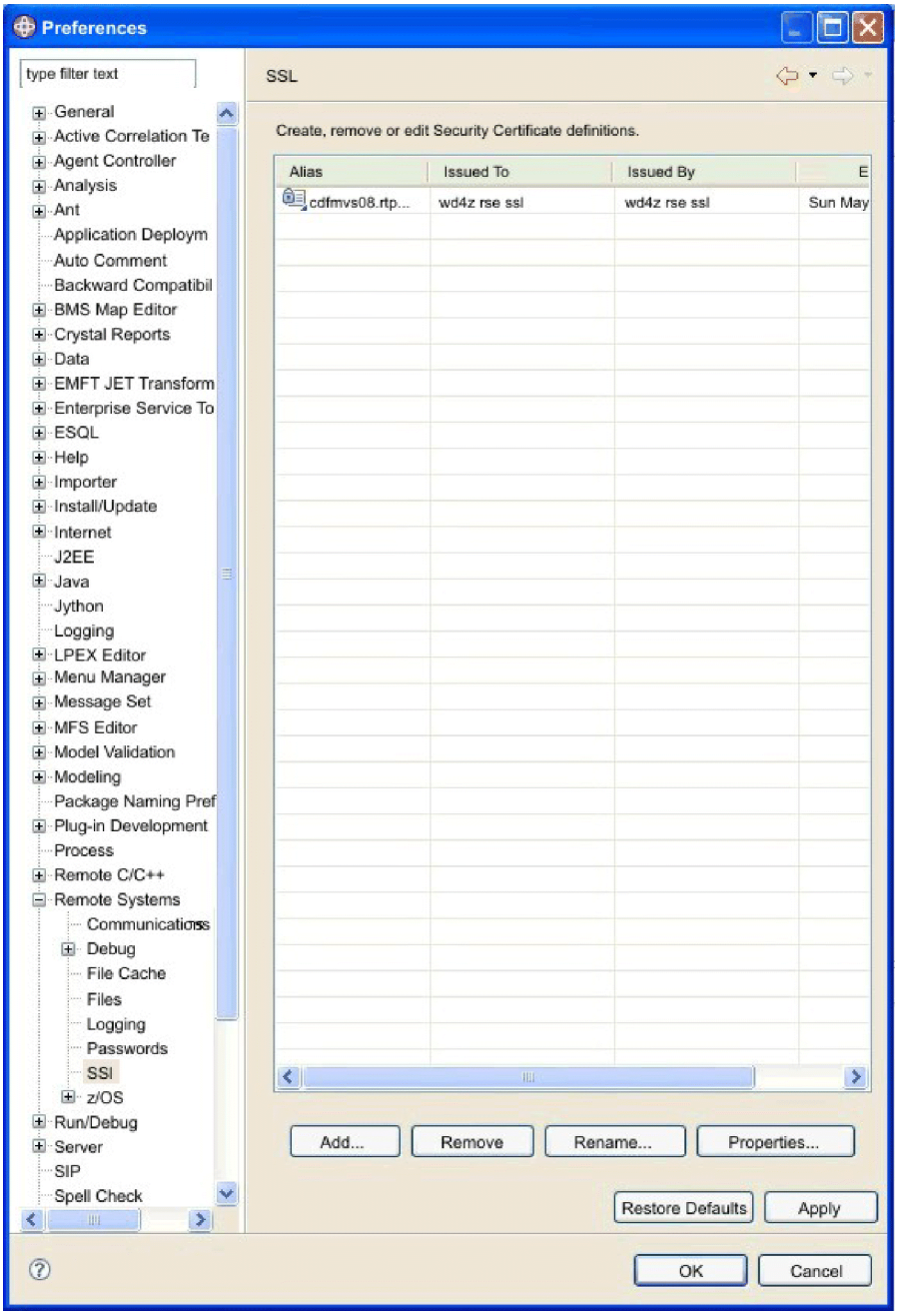


Figure 27. Preferences dialog - SSL

If encrypted 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 131 and "RSE user logging" on page 132.

(Optional) Enable FIPS 140-2 compliancy

RSE supports FIPS 140-2 compliant encrypted communication, which is disabled by default. Using encrypted communication is a prerequisite for this function. Take the following steps to enable FIPS 140-2 compliant encrypted communication:

1. Specify `GSK_FIPS_STATE=ON` in `rse.env`.

2. Restart the RSE started task to pick up the change.

(Optional) Add X.509 client authentication support

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

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, `keyring.racf`, so that it is available to validate the client certificates.

   ```
   RACDCERT ID(stcrse) CONNECT(CERTAUTH LABEL('HighTrust CA') +
     RING(keyring.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.

Manage encryption protocols and ciphers

RSE daemon and RSE server rely on cryptographic services provided by other products (System SSL and Java cryptographic services). These services support multiple communication protocols and multiple encryption ciphers, all of which must be enabled before they can be used. By default, RSE uses the service's defaults for protocol and cipher enablement, with the exception that encryption ciphers which are known to be insecure are automatically disabled. z/OS Explorer client also relies on the defaults of Java cryptographic services.

z/OS Explorer allows you to overrule the defaults for protocol and cipher enablement, with exception of the known insecure ciphers, which are always disabled. Note that the actual protocol and cipher selection is done during the handshake between client and host, and cannot be controlled directly.

Managing encryption ciphers

z/OS Explorer allows you to specify System SSL variable `GSK_V3_CIPHER_SPECS` in `rse.env`. This variable specifies the encryption cipher selection specifications in order of preference as a string consisting of one or more 2-character values. RSE daemon will disable ciphers that are known to be insecure (if present), and pass this selection on to RSE server to be used by Java cryptographic services.

For example:

```
GSK_V3_CIPHER_SPECS=3536372F30310A100D0F0C
```

Specifies that the cipher with ID `35` (256-bit AES encryption with `SHA-1` message authentication and RSA key exchange) is the preferred cipher. It will be enabled if not already enabled by default. Cipher ID `36` is next in line, followed by cipher ID `37`, and so on. For a list of supported ciphers and their 2-character ID, see *Cryptographic Services System SSL Programming (SC24-5901)*.

Managing encryption protocols

z/OS Explorer allows you to specify System SSL variables `GSK_PROTOCOL_*` in `rse.env`. These variables specify whether the specified encryption protocol is enabled. RSE daemon will pass this selection on to RSE server to be used by Java cryptographic services. Note that at the time of publication, z/OS Explorer clients use the TLSv1.0 protocol, and all clients require changes to use another protocol.

For example:

```
GSK_PROTOCOL_SSLV3=OFF
GSK_PROTOCOL_TLSV1=ON
```

Explicitly disables the SSLv3.0 protocol, and explicitly enables the TLSv1.0 protocol. For a list of supported protocols and the matching variable names, see *Cryptographic Services System SSL Programming (SC24-5901)*.

Support for SSLv3 (deprecated)

Due to a vulnerability in the SSLv3 (Secure Socket Layer) protocol, support for this protocol is deprecated in z/OS Explorer. However, SSL was the default protocol up until the deprecation, which implies that existing host and client setups require updates to switch to TLS (Transport Layer Security).

The recommended action for the host is to explicitly disable the usage of SSL by adding `GSK_PROTOCOL_SSLV3=OFF` to `rse.env` (support for TLS is already enabled by default).

During the transition period in which older clients are updated to use TLS, the z/OS Explorer host must be able to support both SSL and TLS. Depending on your service level of System SSL and Java, this might be a more involved process, as outlined here:

1. Copy Java's `lib/security/java.security` to `/etc/zexpl/ssl.java.security` and comment out the `jdk.tls.disabledAlgorithms=SSLv3` line. This step must be repeated each time you apply service to Java.

   ```
   $ cp /usr/lpp/java/J7.0/lib/security/java.security /etc/zexpl/ssl.java.security
   $ oedit /etc/zexpl/ssl.java.security
         -> comment: #jdk.tls.disabledAlgorithms=SSLv3
   ```

2. Add the following statements to the end of `rse.env` (note the double equal sign (`==`) in the `java.security.properties` line) and restart RSE daemon to pick up the changes.

```
GSK_PROTOCOL_SSLV3=ON
GSK_PROTOCOL_TLSV1=ON
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dcom.ibm.jsse2.disableSSLv3=false"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Djava.security.properties==/etc/zexpl/ssl.java.security"
```

Chapter 13. Setting up TCP/IP

This section 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, z/OS Explorer 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

executed on CDFMVS08 -- Wed  Jul   2  13:11:54  EDT  2008
executed by uid=1(USERID) gid=0(GROUP)
RSE_CFG=/etc/zexpl
RSE_HOME=/usr/lpp/IBM/zexpl
-- /usr/lpp/IBM/zexpl/bin/FEK.init.env
-- /usr/lpp/IBM/zexpl/bin/rse.product.env
-- /etc/zexpl/rse.env
-- /usr/lpp/IBM/zexpl/bin/FEK.final.env

current address space size limit is 1902092288 (1814.0 MB)
maximum address space size limit is 2147483647 (2048.0 MB)

------------------------------------------------------------
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 158). 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 158).
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 z/OS Explorer

As stated before, z/OS Explorer 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.

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 158, 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 43 on page 162 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 159, 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 43 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 43. 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 43. 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 43 on page 162 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* (SC27-8437). 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 `rse.env`, where `<resolver file>` or `<resolver data>` represents the name of your IP resolver file:

```
RESOLVER_CONFIG=<resolver file>
```

or

```
RESOLVER_CONFIG='<resolver data set>'
```

Appendix. Accessibility features for z/OS Explorer

Accessibility features assist users who have a disability, such as restricted mobility or limited vision, to use information technology content successfully.

Overview

z/OS Explorer includes the following major accessibility features:

- Keyboard-only operation
- Operations that use a screen reader
- Color and typeface preferences

z/OS Explorer uses IBM Installation Manager to install the product. You can read about the accessibility features for IBM Installation Manager in IBM Installation Manager documentation.

z/OS Explorer uses the latest W3C Standard, WAI-ARIA 1.0, to ensure compliance with US Section 508 and Web Content Accessibility Guidelines (WCAG) 2.0. To take advantage of accessibility features, use the latest release of your screen reader and the latest web browser that is supported by z/OS Explorer.

The z/OS Explorer online product documentation in IBM Knowledge Center is enabled for accessibility. The accessibility features of IBM Knowledge Center are described in the Accessibility section of the IBM Knowledge Center help.

Keyboard navigation

You can use keyboard shortcuts to navigate the help system and the product without using a mouse. For more information, see the *Keyboard shortcuts for the help system in the product* topic in z/OS Explorer documentation.

Interface information

The z/OS Explorer online product documentation is available in IBM Knowledge Center, which is viewable from a standard web browser.

PDF files have limited accessibility support. With PDF documentation, you can use optional font enlargement, high-contrast display settings, and can navigate by keyboard alone.

To enable your screen reader to accurately read syntax diagrams, source code examples, and text that contains period or comma PICTURE symbols, you must set the screen reader to speak all punctuation.

Related accessibility information

In addition to standard IBM help desk and support websites, IBM has a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

TTY service 800-IBM-3383 (800-426-3383) (within North America)

For more information about the commitment that IBM has to accessibility, see IBM Accessibility.

Bibliography

Referenced publications

The following publications are referenced in this document:

Table 44. Referenced publications

Publication title	Order number	Reference	Reference Web site
Program Directory for IBM Explorer for z/OS	GI13-4314	z/OS Explorer	http://www-01.ibm.com/support/docview.wss?uid=swg27047234
IBM Explorer for z/OS Host Configuration Quick Start Guide	GI13-4313	z/OS Explorer	http://www-01.ibm.com/support/docview.wss?uid=swg27047234
IBM Explorer for z/OS Host Configuration Guide	SC27-8437	z/OS Explorer	http://www-01.ibm.com/support/docview.wss?uid=swg27047234
IBM Explorer for z/OS Host Configuration Reference	SC27-8438	z/OS Explorer	http://www-01.ibm.com/support/docview.wss?uid=swg27047234
IBM Explorer for z/OS Host Configuration Utility Guide	SC27-8436	z/OS Explorer	http://www-01.ibm.com/support/docview.wss?uid=swg27047234
Communications Server IP Configuration Guide	SC31-8775	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Configuration Reference	SC31-8776	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP Diagnosis Guide	GC31-8782	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server IP System Administrator's Commands	SC31-8781	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server SNA Network Implementation Guide	SC31-8777	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Communications Server SNA Operations	SC31-8779	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Cryptographic Services System SSL Programming	SC24-5901	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
DFSMS Macro Instructions for Data Sets	SC26-7408	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
DFSMS Using data sets	SC26-7410	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
JES2 Initialization and Tuning Guide	SA22-7532	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
JES3 Initialization and Tuning Guide	SA22-7549	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Language Environment Customization	SA22-7564	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Language Environment Debugging Guide	GA22-7560	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/

Table 44. Referenced publications (continued)

Publication title	Order number	Reference	Reference Web site
MVS Callable Services for HLL	SA22-7613	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Diagnosis: Tools and Service Aids	GA22-7589	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Initialization and Tuning Guide	SA22-7591	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Initialization and Tuning Reference	SA22-7592	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS JCL Reference	SA22-7597	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Planning APPC/MVS Management	SA22-7599	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS Planning Workload Management	SA22-7602	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
MVS System Commands	SA22-7627	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Security Server RACF Command Language Reference	SA22-7687	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Security Server RACF Security Administrator's Guide	SA22-7683	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
TSO/E Customization	SA22-7783	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
TSO/E REXX Reference	SA22-7790	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services Command Reference	SA22-7802	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services Planning	GA22-7800	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
UNIX System Services User's Guide	SA22-7801	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Using REXX and z/OS UNIX System Services	SA22-7806	z/OS 1.13	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
Java™ Diagnostic Guide	SC34-6650	Java 6.0	http://www.ibm.com/developerworks/java/jdk/diagnosis/
Java SDK and Runtime Environment User Guide	/	Java 6.0	http://www-03.ibm.com/servers/eserver/zseries/software/java/

The following Web sites are referenced in this document:

Table 45. Referenced Web sites

Description	Reference Web site
z/OS Explorer IBM Knowledge Center	http://www-01.ibm.com/support/knowledgecenter/SSBDYH/welcome.html
z/OS Explorer library page	http://www-01.ibm.com/support/docview.wss?uid=swg27047051

Table 45. Referenced Web sites (continued)

Description	Reference Web site
z/OS Explorer product page	http://www-01.ibm.com/software/htp/cics/ibmexplforzos/
z/OS Explorer download page	https://developer.ibm.com/mainframe/
z/OS Explorer Recommended service	http://www-01.ibm.com/support/docview.wss?rs=2294 &context=SS2QJ2&uid=swg27006335
z/OS internet library	http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/
IBM Tivoli® Directory Server	http://www-01.ibm.com/software/tivoli/products/directory-server/
Java security information	http://www.ibm.com/developerworks/java/jdk/security/
CA support home page	https://support.ca.com/

Informational publications

The following publications can be helpful in understanding setup issues for the requisite host system components:

Table 46. Informational publications

Publication title	Order number	Reference	Reference website
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 Java EE standard. Java EE 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 Java EE, 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

Notices

This information was developed for products and services offered in the U.S.A.

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 character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

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

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

IBM may use or distribute any of the information you 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 Silicon Valley Lab 555 Bailey Avnue San Jose, CA 95141-1003 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.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1992, 2017.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

Trademarks

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

Terms and conditions for product documentation

Applicability

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

Personal use

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

Commercial use

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

Rights

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

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

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

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

Copyright license

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`.

Adobe and PostScript are trademarks of Adobe Systems Incorporated.

Cell Broadband Engine - Sony Computer Entertainment Inc.

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation, in the United States, other countries, or both.

Intel, Intel Centrino, Intel SpeedStep, Intel Xeon, Celeron, Itanium, and Pentium are trademarks of Intel Corporation in the United States, or other countries, or both.

IT Infrastructure Library is a trademark of Central Computer and Telecommunications Agency

ITIL is a trademark of The Minister for the Cabinet Office

Linear Tape-Open, LTO, and Ultrium are trademarks of HP, IBM Corp., and Quantum

Linux is a trademark of Linus Torvalds

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

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Readers' Comments — We'd Like to Hear from You

IBM Explorer for z/OS
Host Configuration Reference Guide

Publication No. SC27-8438-02

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
SC27-8438-02

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

Cut or Fold
Along Line

Printed in USA

SC27-8438-02