Rational Developer for System z v7.5.1 Release Notes - Host

The following topics provide information about enhancements to the Rational Developer for System z Host in Version 7.5.1:

ISPF prerequisite

The following table updates the prerequisite table for ISPF with the PTF numbers for requisite ISPF APARs. This list replaces previous versions in the Host Configuration Release Notes (technote), the Host Readme (CD), the Prerequisite Guide (CD), the Host Planning Guide (CD) and the Program Directory (CD).

Program Number	Product Name	PTFs or Service Levels Required
5694-A01	ISPF (z/OS v 1.10)	APAR OA24881 (enhance TSO/ISPF Client Gateway support) PTF UA43753 APAR OA27190 (minor bug fixes) PTF UA45032 APAR OA27174 (fix bug that can crash RSE server) PTF UA45333 APAR OA27721 (minor bug fixes) no PTF number known yet
5694-A01	ISPF (z/OS v 1.9)	APAR OA24403 (add TSO/ISPF Client Gateway support) PTF UA41312 APAR OA24882 (enhance TSO/ISPF Client Gateway support) PTF UA43752 APAR OA27190 (minor bug fixes) PTF UA45031 APAR OA27174 (fix bug that can crash RSE server) PTF UA45332 APAR OA27721 (minor bug fixes) no PTF number known yet
5694-A01	ISPF (z/OS v 1.8)	APAR OA20345 (nested command output and enhance BUILD) PTF UA33575 APAR OA20449 (add NESTMACS support) PTF UA34052 APAR OA24403 (add TSO/ISPF Client Gateway support) PTF UA41311 APAR OA24882 (enhance TSO/ISPF Client Gateway support) PTF UA43751 APAR OA27190 (minor bug fixes) PTF UA45030 APAR OA27174 (fix bug that can crash RSE server) PTF UA45242 APAR OA27721 (minor bug fixes) no PTF number known yet

Parent topic: Rational Developer for System z v7.5.1 Host Readme

Console messages

The following list of new console messages can be generated by the lock daemon, RSE daemon or RSE server:

- FEK004I = RseDaemon: Max Heap Size={0}MB and private AS Size={1}MB
- FEK143E = gsk_attribute_set_enum(GSK_CLIENT_AUTH_TYPE) failed. reason=({0})
- FEK144E = gsk_get_cert_info failed. reason=({0})
- FEK145E = gsk_secure_socket_read() failed. reason=({0})
- FEK146E = gsk_secure_socket_write() failed. reason=({0})
- FEK209I = No Process to be displayed
- FEK501I = Lock daemon started, port={0}, cleanup interval={1}, log level={2}
- FEK502I = Lock daemon terminating
- FEK510E = Lock daemon, missing port
- FEK511E = Lock daemon, wrong port, port={0}
- FEK512E = Lock daemon, socket error, port={0}
- FEK513W = Lock daemon, registration failed, ASID={0}, TCB={1}, USER={2} \
- FEK514W = Lock daemon, wrong log level, log level={0}
- BPXM023I (stclock) dataset[(member)] NOT LOCKED
- BPXM023I (stclock) dataset[(member)] LOCKED BY userid
- BPXM023I (stclock) command, WRONG COMMAND
- BPXM023I (stclock) command, MISSING ARGUMENT
- BPXM023I (stclock) argument, WRONG ARGUMENT

Parent topic: Rational Developer for System z v7.5.1 Host Readme

JES Job Monitor

Several new features have been added to JES Job Monitor, which have not been documented yet in the *Rational Developer for System z Host Configuration Guide* (SC23-7658).

Embedded LE options

FEJJCNFG, JES Job Monitor configuration file

Show JCL command

Parent topic: Rational Developer for System z v7.5.1 Host Readme

Embedded LE options

The following LE options are embedded in the JES Job Monitor load module:

- ALL31(ON)
- HEAP(32K,32K,ANY,KEEP,8K,4K)
- POSIX(ON)
- STACK(128K,128K,ANYWHERE,KEEP)
- THREADHEAP(4K,4K,ANY,KEEP)
- THREADSTACK(OFF)
- STORAGE(NONE,NONE,NONE,0)

If required, these options can be changed by providing alternate values in the startup JCL. The sample JCL below shows an updated version of the startup JCL, `FEK.SFEKSAMP(FEJJCL)`, where parameter `LEPRM` can be used to specify LE options.

```
//*
//* JES JOB MONITOR
//*
//JMON     PROC PRM=,                   * PRM='-TV' TO START TRACING
//            LEPRM='RPTOPTS(ON)',
//            HLQ=FEK,
//            CFG=FEK.#CUST.PARMLIB(FEJJCNFG)
//*
//JMON     EXEC PGM=FEJJMON,REGION=0M,TIME=NOLIMIT,
//            PARM=('&LEPRM,ENVAR("_CEE_ENVFILE=DD:ENVIRON")/&PRM')
//STEPLIB  DD DISP=SHR,DSN=&HLQ..SFEKAUTH
//ENVIRON  DD DISP=SHR,DSN=&CFG
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//         PEND
//*
```

Note: Use caution when changing LE options, as inappropriate values can stop JES Job Monitor from working as designed.

Parent topic: JES Job Monitor

FEJJCNFG, JES Job Monitor configuration file

New, optional, control directives for JES Job Monitor have been added to the FEJJCNFG configuration file.

Directive	Default value
APPLID = application_id	FEKAPPL
CONSOLE_NAME = {console_name | &SYSUID}	JMON
GEN_CONSOLE_NAME = {ON | OFF}	OFF

APPLID

Specifies the application identifier used for identifying JES Job Monitor to your security software. The default is FEKAPPL . Uncomment and change to the desired application ID.

Note: This value must match the application ID set for RSE in the rsed.envvars configuration file. If these values differ, RSE cannot connect the client to JES Job Monitor.

CONSOLE_NAME

Specifies the name of the EMCS console used for issuing commands against jobs (Hold, Release, Cancel and Purge). The default is JMON . Uncomment and change to the desired console name, using the following guidelines:

- CONSOLE_NAME must be either a console name consisting of 2 to 8 alphanumeric characters, or '&SYSUID' (without quotes).
- If a console name is specified, a single console by that name is used for all users. If the console by that name happens to be in use, then the command issued by the client will fail.
- If &SYSUID is specified, the client user ID is used as the console name. Thus a different console is used for each user. If the console by that name happens to be in use (for example, the user is using the SDSF ULOG), then the command issued by the client might fail, depending on the GEN_CONSOLE_NAME setting.

No matter which console name is used, the user ID of the client requesting the command is used as the LU of the console, leaving a trace in syslog messages IEA630I and IEA631:

```
IEA630I OPERATOR console NOW ACTIVE, SYSTEM=sysid, LU=id
IEA631I OPERATOR console NOW INACTIVE, SYSTEM=sysid, LU=id
```

Note: If you change the console name, you must only provide matching security definitions for the MVS.MCSOPER.console profile in the OPERCMDS class, where console is the actual console name. The other references to JMON in the security setup, CONSOLE class and PERMIT WHEN(CONSOLE(JMON)) must NOT be changed.

GEN_CONSOLE_NAME

Enables or disables automatic generating of alternative console names. The default is OFF . Uncomment and change to ON to enable alternative console names.

This directive is only used when CONSOLE_NAME=&USERID and the user ID is not available as console name.

If GEN_CONSOLE_NAME=ON, an alternative console name is generated by appending a single numeric digit to the user ID. The digits 0 through 9 are attempted. If no available console is found, the command issued by the client fails.

If `GEN_CONSOLE_NAME=OFF`, the command issued by the client fails.

Parent topic: JES Job Monitor

Show JCL command

Similar to the SDSF **SJ** action character, JES Job Monitor 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.

Authorization for the Show JCL command is controlled by the `LIMIT_COMMANDS` directive in `FEJJCNFG` , and profiles in the `JESSPOOL` class of your security product.

	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 table below.

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

Use the following substitutions in the table above:

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`. The behavior is identical when `JESSPOOL` is active, since the class, by default, denies permission if a profile is not defined.

Note that Show JCL is not an operator command like the other JES Job Monitor commands (Hold, Release,

Cancel and Purge), so no profile is needed in the `OPERCMDS` class.

Parent topic: JES Job Monitor

RSE Lock daemon

By switching to a single server setup, where multiple users are assigned to a single thread pool server, RSE lost the ability to track who owns a lock on a dataset or member. System commands stop at address space level, which is the thread pool server.

To address this problem, a new server has been created, the lock daemon. It can track all dataset/member locks done by RSE users, as well as locks done by other products, like ISPF.

Note: The Rational Developer for System z client will display the lock information where needed if the client is at a matching service level.

Parent topic: Rational Developer for System z v7.5.1 Host Readme

Setup

Customize and submit FEK.SFEKSAMP(FEKSET01)

The `FEKSET01` job will create a sample lock daemon started task JCL in `FEK.#CUST.PROCLIB(LOCKD)` for customization purposes. The original version is available as `FEK.SFEKSAMP(FEKLOCKD)`.

```
//*
//* RSE LOCK DAEMON
//*
//LOCKD    PROC HOME='/usr/lpp/rdz',
//            CNFG='/etc/rdz',
//            LOG=1
//*
//LOCKD    EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//            PARM='PGM &HOME/bin/lockd.sh &CNFG &LOG'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//         PEND
//*
```

The `FEKSET01` job also updates the active `/etc/rsed.envvars` and adds the following lock daemon related statements to the end of the file:

```
_RSE_LOCKD_PORT=4036
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dlock.daemon.port=$_RSE_LOCKD_PORT"
_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dlock.daemon.cleanup.interval=1440"
_RSE_LOCKD_CLASS=com.ibm.ftt.rse.mvs.server.miners.MVSLockDaemon
```

`_RSE_LOCKD_PORT` Specifies the port used by the lock daemon. The default port is `4036`. Communication on this port is confined to the host. The remaining variables should not be modified.

PROCLIB updates

Customize `FEK.#CUST.PROCLIB(LOCKD)` and copy the member to `SYS1.PROCLIB`, or other valid PROCLIB data set.

PARMLIB updates

The lock daemon must be started before users log on to the RSE daemon so it can track the lock requests by these users. Therefore it is advised to start the lock daemon at system startup, for example by adding a start command to `SYS1.PARMLIB(COMMNDxx)`.

Security settings

The lock daemon does not require any special security permits. For restricted environments, the server needs:

- read/execute permission to Java libraries
- read/execute permission to `/usr/lpp/rdz/*`
- read permission to `/etc/rdz/*`
- read permission to any data set in the `rsed.envvars` STEPLIB concatenation
- read permission to the `PROGRAM` class profile protecting `SYS1.LINKLIB` (the samples in the *Rational Developer for System z Host Configuration Guide* (SC23-7658) use profile `**`).

When used as started task, the server must be defined to your security software

```
LISTUSER STCLOCK OMVS
ADDUSER STCLOCK NOPASSWORD DFLTGRP(STCGROUP) +
  OMVS(UID(9) HOME(/tmp) PROGRAM(/bin/sh)) +
  NAME('RDZ LOCK DAEMON') +
  DATA('RATIONAL DEVELOPER FOR SYSTEM Z')

RLIST STARTED LOCKD.* ALL STDATA
RDEFINE STARTED LOCKD.* +
  STDATA(USER(STCLOCK) GROUP(STCGROUP) TRUSTED(NO)) +
  DATA('RDZ LOCK DAEMON')
SETROPTS RACLIST(STARTED) REFRESH
```

Parent topic: RSE Lock daemon

Usage

How it works

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

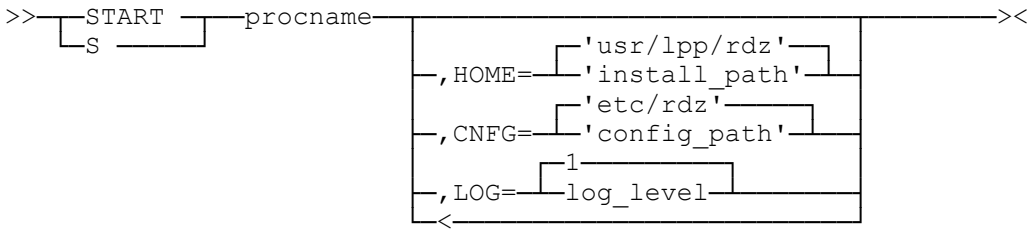
Note that registration is done at connect time only, so all RSE users active before the lock daemon was (re)started will not be registered.

When the lock daemon receives a dataset query, it scans the system's Global Resource Serialization (GRS) queues. If the ASID and TCB match that of a registered user, the user ID is returned as lock owner. Otherwise the jobname/user ID related to the ASID is returned as lock owner.

A console message with the registration info is displayed if the registration fails, so that an operator can issue the **D GRS,RES=(*,dataset[(member)])** operator command and manually match the ASID and TCB values with the one reported by the system as lock owner.

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

Operator commands



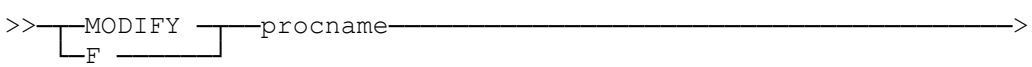
procname
: The name of the member in a procedure library that is used to start the server. The default name used during the host configuration is `LOCKD` .

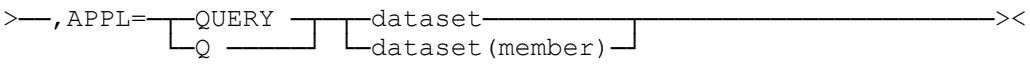
HOME='install_path'
: Path prefix and the mandatory `/usr/lpp/rdz` used to install Developer for System z. The default is `'/usr/lpp/rdz'`. Note that the z/OS UNIX path is case sensitive and that it must be enclosed in single quotes (') to preserve lowercase characters.

CNFG='config_path'
: Absolute location of the configuration files stored in z/OS UNIX. The default is `'/etc/rdz'`. Note that the z/OS UNIX path is case sensitive and that it must be enclosed in single quotes (') to preserve lowercase characters.

LOG=log_level
: The detail level of output in DD STDOUT.

- 0 : Log error messages only.
- 1 : Log error and warning messages (default).
- 2 : Log error, warning and informational messages.





procname

The name of the member in a procedure library that is used to start the server. The default name used during the host configuration is `LOCKD` .

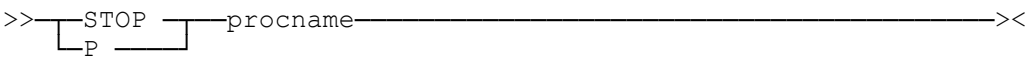
QUERY dataset[(member)]

Query the lock status of the listed data set or member. The server will reply with one of the following messages:

```
BPXM023I (stclock) dataset[(member)] NOT LOCKED
BPXM023I (stclock) dataset[(member)] LOCKED BY userid
```

Notes:

- The server will also report locks held by other products, like ISPF.
- Locks held by RSE clients started before the lock daemon will result in the thread pool server address space (`RSEDx`) being reported as lock owner.



procname

The name of the member in a procedure library that is used to start the server. The default name used during the host configuration is `LOCKD`.

Parent topic: RSE Lock daemon

Enhanced client authentication support

Rational Developer for System z supports multiple ways to authenticate a user ID provided by a client upon connection. Note that the authentication data provided by the client is only used once, during initial connection setup. Once a user ID is authenticated, the user ID and self-generated PassTickets are used for all actions that require authentication. Also note that SSL encryption is supported for each authentication method, and it is required for certificate authentication.

User ID and password

User ID and one-time password

X.509 Certificate

Certificate validation

Certificate authentication

JES Job Monitor authentication

Sample setup for supporting X.509 certificate logons

Parent topic: Rational Developer for System z v7.5.1 Host Readme

User ID and password

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

No additional customization is needed to use this authentication mechanism.

Parent topic: Enhanced client authentication support

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 since 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.

No additional customization is needed to use this authentication mechanism.

Parent topic: Enhanced client authentication support

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.

Developer for System z will require additional customization, as described in Certificate validation and Certificate authentication. Also note that this authentication method is only supported by the RSE daemon connection method, and that SSL must be enabled.

Parent topic: Enhanced client authentication support

Certificate validation

RSE daemon starts certificate processing by validating the 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 be consulted also. After this validation, the certificate is processed for the authentication step, as described in Certificate authentication.

RSE daemon uses z/OS System SSL services to validate the certificate. Therefore, ensure that `SYS1.SIEALNKE` is accessible by the RSE daemon, either via LPA, LINKLIST or via the STEPLIB directive in `rsed.envvars`.

Also ensure that `ssl.properties` is set up correctly (SSL must be enabled), as described in the *Rational Developer for System z Host Configuration Guide* (SC23-7658).

Developer for System z v7510 maintenance note

Certificate Authority (CA) validation

(Optional) Query a Certificate Revocation List (CRL)

Parent topic: Enhanced client authentication support

Developer for System z v7510 maintenance note

X.509 Certificate support is added in v7510 of Rational Developer for System z. This maintenance level also introduces a new directive to `ssl.properties`:

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

If the `server_keystore_type` value equals `JKS` (the default), a key store, created by the Java **keytool** program, is used to store the RSE server certificate.

If the `server_keystore_type` value equals `JCERACFKS`, a key ring, created by your security product, is used to store the RSE server certificate.

The RSE server certificate is the certificate RSE server uses to authenticate itself to the client. When using a key ring to store the certificate, the RSE server certificate can be the same as the RSE daemon certificate, thus simplifying the setup.

Note that using a key ring to store the RSE server certificate is currently not described in the *Rational Developer for System z Host Configuration Guide* (SC23-7658). The publication only describes the Java key store method for RSE server. However, the actions needed to use a key ring are described in the RSE daemon section of the SSL setup, as this one supported key rings before. It is also briefly documented below, in Sample setup for supporting X.509 certificate logons.

Notes:

1. RSE daemon also supports the usage of a **gskkyman** key database to store certificates. This is NOT supported for the RSE server.
2. The RSE server certificate MUST be the first certificate in the Java key store or SAF key ring, and the CA certificate(s) must be added afterwards. This is due to current limitations where RSE server is unable to select a specific certificate. RSE server always pulls the first certificate. The System SSL validation routines used by RSE daemon are able to access all the certificates.

Parent topic: Certificate validation

Certificate Authority (CA) validation

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

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

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

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

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

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

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

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

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

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

Attention: If you rely on RSE daemon instead of your security software to authenticate a user, you must be cautious not to mix CA's with a TRUST and HIGHTRUST status. 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. See Authentication by RSE daemon for more information on using RSE daemon for the authentication step.

Parent topic: Certificate validation

(Optional) Query a Certificate Revocation List (CRL)

If desired, you can instruct RSE daemon to check one or more Certificate Revocation List(s) (CRL) to add extra security to the validation process.

This is done by adding CRL related environment variables to `rsed.envvars`. The following sample definitions are available in `rsed.envvars`:

GSK_CRL_SECURITY_LEVEL
Specifies the level of security SSL applications will use when contacting LDAP servers to check CRLs for revoked certificates during certificate validation. The default is `MEDIUM`. Uncomment and change to enforce the usage of the specified value. The following values are valid:

- `LOW` - Certificate validation will not fail if the LDAP server cannot be contacted.
- `MEDIUM` - Certificate validation requires the LDAP server to be contactable, but does not require a CRL to be defined. This is the default.
- `HIGH` - Certificate validation requires the LDAP server to be contactable and a CRL to be defined.

Note: This directive requires z/OS 1.9 or higher.

GSK_LDAP_SERVER
Specifies one or more blank-separated LDAP server host names. Uncomment and change to enforce the usage of the specified LDAP servers to obtain their CRL.

The host name can either be a TCP/IP address or an URL. Each host name can contain an optional port number separated from the host name by a colon (:).

GSK_LDAP_PORT
Specifies the LDAP server port. The default is `389`. Uncomment and change to enforce the usage of the specified value.

GSK_LDAP_USER
Specifies the distinguished name to use when connecting to the LDAP server. Uncomment and change to enforce the usage of the specified value.

GSK_LDAP_PASSWORD
Specifies the password to use when connecting to the LDAP server. Uncomment and change to enforce the usage of the specified value.

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

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

Parent topic: Certificate validation

Certificate authentication

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

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

`enable.certificate.mapping` is a new directive in `rsed.envvars`.

```
#_RSE_JAVAOPTS="$_RSE_JAVA_OPTS -Denable.certificate.mapping=false"
```

Use your security software to authenticate a logon with a X.509 certificate. The default is `true`. Uncomment this option to have RSE daemon do the authentication without relying on the X.509 support of your security software.

Authentication by your security software

Authentication by RSE daemon

Parent topic: Enhanced client authentication support

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 you 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, it 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 if there is a matching certificate name filter defined in the `DIGTNMAP` and/or `DIGTCRIT` classes. If so, it returns the user ID associated with the most specific matching filter.

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

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

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

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

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

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

Refer to *Security Server RACF Security Administrator's Guide* (SA22-7683) for more information on X.509 certificates, how they are managed by RACF, and how to define certificate name filters. Refer to *Security Server RACF Command Language Reference* (SA22-7687) for more information on the **RACDCERT** command.

Authentication by RSE daemon

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

This function is intended to be used if your security product does not support authenticating a user based upon a X.509 certificate, or if your certificate would fail the test(s) done by your security product (for example, it 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.

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, as RSE daemon cannot check if the one who signed the client certificate is highly trusted or just trusted. See Certificate validation for more information on accessible CA certificates.

Parent topic: Certificate authentication

JES Job Monitor authentication

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

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 that:

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

Parent topic: Enhanced client authentication support

Sample setup for supporting X.509 certificate logons

Perform the following steps to allow logons using X.509 certificates. This sample setup uses RACF to store the certificates. Note that the RACF steps must be performed by a security administrator.

1. Define the initial certificate related permits to allow the RSE user ID, STCRSE , access to his RACF key ring.

   ```
   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
   ```

2. Create a self-signed certificate, with label `rdzrse`, to identify the RSE host (both daemon and server) to the client.

   ```
   RACDCERT ID(stcrse) GENCERT SUBJECTSDN(CN('rdz rse ssl') +
     OU('rdz') O('IBM') L('Raleigh') SP('NC') C('US')) +
     NOTAFTER(DATE(2017-05-21)) WITHLABEL('rdzrse') +
     KEYUSAGE(HANDSHAKE)
   ```

3. Add the RSE host certificate to a newly created key ring, `rdzssl.racf`. Note that if you use a host certificate signed by a Certificate Authority (CA), the CA certificate must also be added to the key ring.

   ```
   RACDCERT ID(stcrse) ADDRING(rdzssl.racf)
   RACDCERT ID(stcrse) CONNECT(LABEL('rdzrse') RING(rdzssl.racf) +
     DEFAULT USAGE(PERSONAL))
   ```

   This concludes the RACF setup for the RSE host certificate.

4. 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, as is it used for the certificate authentication part of the logon process.

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

5. Add the CA certificate to the key ring, `rdzssl.racf`, so that it's available to validate the client certificates.

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

   This concludes the RACF setup for the CA certificate.

6. 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)
   ```

7. 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)
   ```

8. 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
```

or

```
SETROPTS RACLIST(SERVAUTH) REFRESH
```

This concludes the RACF setup for the HostIdMappings extension.

9. Update `/etc/rdz/ssl.properties` so RSE will know to use SSL encrypted communication with the client.

```
enable_ssl=true

# Daemon Properties
daemon_keydb_file=rdzssl.racf
daemon_key_label=rdzrse

# Server Properties
server_keystore_file=rdzssl.racf
server_keystore_type=JCERACFKS
```

This concludes the RSE configuration setup for SSL.

10. Restart the RSE started task to start accepting client logons using X.509 certificates.

Parent topic: Enhanced client authentication support