The Rational solution for Collaborative Lifecycle Management

IBM

Installation Guide for z/OS

Version 4.0.6

The Rational solution for Collaborative Lifecycle Management

Installation Guide for z/OS

Version 4.0.6

Note

Before using this information and the product it supports, read the information in Appendix A, “Notices,” on page 113.

This edition applies to version 4.0.6 of the Rational solution for Collaborative Lifecycle Management products and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Installing on z/OS

Planning to install on z/OS systems

Installing on a z/OS® system has several significant differences from installing on other systems:

- IBM® Installation Manager is not compatible with installations on z/OS systems.
- Installations on z/OS systems are handled by SMP/E (System Modification Program/Extended).
- Jazz™ Team Server and the other CLM applications on z/OS systems use configuration and working directories that are separate from installation directories.
- z/OS installations can be a hybrid of applications installed on z/OS systems and other systems.

For instructions to install the Rational® solution for CLM on a z/OS system, see the following list of topics.

SMP/E installation process

IBM System Modification Program/Extended (SMP/E) installs software and software changes on your z/OS systems.

SMP/E is the tool for installing and maintaining software on z/OS systems. SMP/E controls software changes at the element level. Job Control Language (JCL) is submitted through 3270 emulation, which is also known as "green screen". This method is a standard installation method that is used by IBM WebSphere® Application Server for z/OS, DB2® for z/OS, and other z/OS products.

The Rational solution for Collaborative Lifecycle Management (CLM) components for z/OS are provided in a compressed file that contains program directories and binary files:

Program directories (PDF files)

The program directories contain the instructions for installing the V4.0.6 CLM applications on z/OS: Rational Team Concert, Rational Quality Manager, and Rational Requirements Composer. Each program directory is specific to each product, but follows a template. The program directories contain standard instructions for installing Jazz Team Server and all of the common files on z/OS by using SMP/E. Use one of the program directories to complete the installation. The program directories are also available online. For more information, see The Rational Solution for CLM V4.0.6 program directories for z/OS at http://www.ibm.com/support/docview.wss?&uid=swg27041020.

Remember: The program directories list specific directory path names. If you extract your files to a different location, you must update the path names to match your location.

Binary files

The binary files are the files to be uploaded to z/OS for the SMP/E installation process.

Restrictions:

- No compressed files can be installed directly to z/OS without SMP/E.
- The binary files are not useful outside of SMP/E. The program directories provide detailed steps to upload the files and configure provided sample JCL to install by using SMP/E.
- The SMP/E installation process does not include configuration.
- The program directories describe installation only.

The SMP/E package installs several MVS™ data sets. Of particular interest are `hlq.SBLZSAMP` and `hlq.SBLZAUTH`, in which `hlq` is the high-level qualifier that you selected during the SMP/E installation. As noted in the program directories, `hlq.SBLZAUTH` must be APF-authorized. Many of the additional configuration steps involve editing and submitting JCL that is found in the `hlq.SBLZSAMP` data set. If you plan to use the Rational Team Concert ISPF client, `hlq.SBLZAUTH` must be available in LNKLST.

The SMP/E package also installs one z/OS UNIX System Services file directory: `/@pathPrefix@/usr/lpp/jazz/v4.0.6`, where `@pathPrefix@` is the path prefix that you specified during the SMP/E installation.

The z/OS installation includes nine FMIDs (Function Modification Identifiers):

HRCC406 - Common components
This FMID is a prerequisite for the other FMIDs and must always be installed. It includes the SMP/E JCL and several utilities that other FMIDs share.

HRJS406 - Jazz Team Server
Install this FMID if you plan to run Jazz Team Server on z/OS. HRJS406 includes the server executable files that are installed to the z/OS UNIX System Services (USS) file system. HRJS406 also includes the startup JCL installed to a z/OS data set.

HRCM406 - Rational Team Concert - Change and Configuration Management (CCM)
Install this FMID if you plan to use the server CCM capabilities of Rational Team Concert on z/OS.

HRQM406 - Rational Quality Manager - Quality Management (QM)
Install this FMID if you plan to use the server QM capabilities of Rational Quality Manager on z/OS.

HRRM406 - Rational Requirements Composer - Requirements Management (RM)
Install this FMID if you plan to use the server RM capabilities of Rational Requirements Composer on z/OS.

HROC406 - Optional components
Install this FMID if you plan to run the server by using Tomcat on z/OS.

HRBT406 - Rational Team Concert - Build System Toolkit
Install this FMID if you plan to run any of the following functions on z/OS:

- Enterprise builds
- Jazz Build Engine
- Enterprise gateway
- ISPF client

- Mass import tool
- Promotion
- Packaging and deployment
- Integration with Rational Developer for System z®
- Context-aware search

HRDV406 - Rational Team Concert - Rational Developer for System z subset
Install this FMID in these cases:
- If you do not have Rational Developer for System z installed and you plan to submit or monitor JCL-based builds on z/OS.
- If you plan to use the Rational Team Concert compilation results view for z/OS compiles as part of your enterprise builds.

HRBA406 - Rational Build Agent
Install this FMID if you plan to run Enterprise Extensions, JCL, or command line builds on z/OS.

Installation process on z/OS

The process to install on z/OS is as follows.
1. Finish the customization on z/OS.
2. If you installed Jazz Team Server on z/OS, configure Jazz Team Server.
3. Configure any CLM applications that you installed on z/OS.
4. Install the Rational Team Concert client on your computer.

Installation information for z/OS system programmers

If you are a system programmer, use this information to supplement the SMP/E installation information.

PARMLIB changes

Use commands to set APF authorizations and modify PARMLIB definitions.

Refer to *MVS Initialization and Tuning Reference* (SA22-7592) for more information about the PARMLIB definitions listed in this section. Refer to *MVS System Commands* (SA22-7627) for more information about sample console commands.

Set z/OS UNIX limits in BPXPRMxx

If you are installing the Rational Team Concert ISPF client, the ISPF daemon requires a large address space size for proper operation. Set one of the following variables:

MAXASSIZE
Set the value of the MAXASSIZE variable in PARMLIB member BPXPRMxx to 2GB, which is the maximum value allowed. MAXASSIZE specifies the maximum address space (process) region size. This is a system-wide limit that is set for all z/OS UNIX address spaces. If you do not want to set a system-wide limit, set the limit for the ISPF client only, as described in the next action.

ASSIZEMAX
Set the value of the ASSIZEMAX variable of the OMVS segment for the

user ID STCISPF or another user ID for the ISPF Daemon started task to 2GB, which provides the ISPF daemon the required region size, regardless of changes to MAXASSIZE.

LNKLST definitions in PROGxx

For the Rational Team Concert ISPF client to operate properly, the library hlq.SBLZAUTH, which contains the BLZPASTK module, must be made available through the LNKLST definition.

LNKLST data sets are defined in PARMLIB member PROGxx, if your site followed IBM recommendations.

The following example shows the required definition:

```
LNKLST ADD NAME(listname) DSNAME(hlq.SBLZAUTH) VOLUME(volser)
```

where *listname* is the name of the LNKLST set being activated, *hlq* is the high-level qualifier you used during SMP/E installation, and *volser* is the volume on which the data set resides if it is not cataloged in the master catalog.

LNKLST definitions can be created dynamically (until the next IPL) using the following console commands,

- `SETPROG LNKLST DEFINE,NAME=LLTMP,COPYFROM=CURRENT`
- `SETPROG LNKLST ADD NAME=LLTMP,DSN=hlq.SBLZAUTH,VOL=volser`
- `SETPROG LNKLST ACTIVATE,NAME=LLTMP`

Run the console command `SETPROG LNKLST,UPDATE,JOB=*` to update an address space so that a specified job or jobs associated with that space can use the current LNKLST set.

APF authorizations in PROGxx

For Job Monitor to access JES spool files, the following must be APF-authorized:

- Module `BLZJMON` in the `hlq.SBLZAUTH` load library, where *hlq* is the high-level qualifier you used during SMP/E installation.
- The Language Environment® (LE) runtime libraries (`CEE.SCEERUN*`)

APF authorizations are defined in `SYS1.PARMLIB(PROGxx)`, if your site follows IBM recommendations.

You can set APF authorizations dynamically with the following console commands:

- `SETPROG APF,ADD,DSN=hlq.SBLZAUTH,SMS`
- `SETPROG APF,ADD,DSN=CEE.SCEERUN,VOL=volser`
- `SETPROG APF,ADD,DSN=CEE.SCEERUN2,VOL=volser`

where *hlq* is the high-level qualifier you used during SMP/E installation, and *volser* is the volume on which the data set resides if it is not SMS-managed.

Note: If the ISPF client is being installed, then the hlq.SBLZAUTH data set must be added to the LNKLST as previously mentioned. If your site automatically APF authorizes LNKLST data sets, then it may not be necessary to implicitly APF authorize the hlq.SBLZAUTH data set.

AUTHPGM definitions in IKJTSOxx

In addition to the load library containing program BLZPASTK being added to the LNKLST, the program itself must be added to the authorized TSO command list in parmlib member IKJTSOxx.

The following example shows the required definition, assuming that IEBCOPY is already in the AUTHPGM list:

```
AUTHPGM NAMES(IEBCOPY,BLZPASTK)
```

TCP/IP ports

Jazz Team Server and the other components on z/OS use the following TCP/IP ports. Notify your firewall and TCP/IP administrators about the ports that are relevant to your installation.

Miscellaneous component ports (non-server ports)

5555	This port is the default port for the Rational Build Forge® Agent. If there are multiple build agents, there are multiple ports.
4152	This port is the default port for the ISPF daemon.
6716	If you use the sample configuration file included with the Rational Team Concert SMP/E package, this port is the default port for the Job Monitor.
@port@	You can specify any available port.

Server ports if the server is deployed on Tomcat

9005	Shutdown port
9080	Non-SSL HTTP/1.1 connector port
9443	SSL HTTP/1.1 connector port
9009	AJP 1.3 connector port
8082	Proxied HTTP/1.1 connector port

WebSphere Application Server ports

Typically, you configure the WebSphere Application Server ports when you configure the application server.

Security on z/OS systems

If you are installing Jazz Team Server and the other CLM applications on z/OS, several tasks are required to make the CLM functions secure and available on z/OS.

These instructions are intended for people who are installing a combination of the Jazz Team Server, any of the CLM applications, the Rational Team Concert™ Build System Toolkit, and the Rational Build Agent on z/OS.

When you are setting up security for an installation on z/OS, include these topics in your planning:

Data set protection

Security is needed for z/OS data sets that are associated with Jazz Team Server, the CLM applications, the Rational Team Concert Build System Toolkit, and Rational Build Agent.

RACF® general resource profiles, GROUPs, and USERs
: Several Resource Access Control Facility (RACF) resources must be configured in order to use the CLM components on z/OS.

UNIX System Services (USS) directory protection
: To install and configure the CLM components on z/OS, you use three main directories and associated subdirectories that need appropriate user and group-level permissions:
  1. Product binaries: Installed by SMP/E, typically to a directory such as `/usr/lpp/jazz/v4.0.6`
  2. Configuration directories: Created by running sample configuration jobs to create and populate a directory such as `/etc/jazz406`
  3. Working directories: Created by running sample configuration jobs to create and populate a directory such as `/u/jazz406`

Database access
: If you are running Jazz Team Server and CLM applications on z/OS, you must provide access from the server to DB2 z/OS databases for the applications and data warehouse.

Two sample members are provided from the *hlq*.`SBLZSAMP` library, where *hlq* is the high-level qualifier that was specified during the SMP/E installation:
- BLZRACF: This sample member is for the server, and is installed with SMP/E FMID HRJS406
- BLZRACFT: This sample member is for the Build System Toolkit and Rational Build Agent, and is installed with SMP/E FMID HRBT406

You can customize these sample members and submit the jobs to perform the RACF updates.

The security considerations for your deployment vary based on which components you installed. Depending on your setup, see one or more of these topics:
- "RACF security on z/OS systems": Read this topic if you have Jazz Team Server or the Build System Toolkit installed on z/OS.
- "Jazz Team Server security on z/OS systems" on page 8: Read this topic if you are deploying Jazz Team Server and the CLM applications on z/OS.
- "Security for the Build System Toolkit and Rational Build Agent on z/OS systems" on page 10: Read this topic if you are deploying the Build System Toolkit or Rational Build Agent on z/OS.

RACF security on z/OS systems

When you deploy Jazz Team Server and the other Rational solution for Collaborative Lifecycle Management (CLM) applications on z/OS systems, you can use several RACF security settings. These settings help secure your data and provide appropriate access to different types of users.

The basic requirements for installing z/OS components are:

SMP/E installation
: Specific z/OS UNIX Systems Services (USS) directories and data sets are created based on which FMIDs are installed. The z/OS data sets are protected based on the data set profiles that you already configured. The user ID of the installer owns the USS directories. Other users have READ and EXECUTE access. You can set up additional security protections.

Creation of USS configuration and working directories
: Running the sample JCL creates configuration and working directories. The

sample jobs are called BLZCP* jobs because there is a version for each component. For example, the Jazz Team Server job is BLZCPJTS, and the Build System Toolkit is BLZCPBTK. You can specify group permissions when you run these sample jobs.

The following sections provide additional security information for installed components:

Data set profile protection

Before you begin the SMP/E installation, create a high-level qualifier (HLQ) for the CLM target and distribution libraries so that you can protect the HLQ by using RACF. Users who work with z/OS functions such as the Rational Team Concert ISPF client or Enterprise Extensions deployment and promotion functions must have READ access on the target data sets. If you copy the target data set elsewhere, users also need READ access on those copy data sets.

If you are installing Jazz Team Server and the CLM applications, you can see an example of RACF statements in the instructions that are provided with the BLZRACF job in *hlq*.SBLZSAMP, where *hlq* is the high-level qualifier that was specified during the SMP/E installation. If you are installing the Rational Team Concert Build System Toolkit, the same RACF commands are provided in the BLZRACFT job in *hlq*.SBLZSAMP.

For most Rational Team Concert data sets, READ access for users and ALTER access for system programmers is sufficient. Ask the system programmer who installed and configured the product for the correct data set names. The default high-level qualifier is BLZ, and a BLZ GROUP is allocated before creating the data set definition. To protect a data set with RACF, the first-level qualifier of the data set name must be a RACF-defined user ID or group name.

These sample RACF commands are included in the JCL:

```
LISTGRP BLZ
  ADDGROUP (BLZ) OWNER(IBMUSER) SUPGROUP(SYS1) -
   DATA('RATIONAL TEAM CONCERT - HLQ STUB')

#  general data set protection
  LISTDSD PREFIX(BLZ) ALL
  ADDSD 'BLZ.**' -
   UACC(READ) DATA('RATIONAL TEAM CONCERT')
  PERMIT 'BLZ.**' -
   CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)

  SETROPTS GENERIC(DATASET) REFRESH

#  show results
  LISTGRP BLZ
  LISTDSD PREFIX(BLZ) ALL
```

User ID OMVS segment creation

You must define a RACF OMVS segment or equivalent that specifies a valid z/OS UNIX user ID (UID), home directory, and shell command for the user who runs the BLZCP* configuration jobs for both the server and the Build System Toolkit. The user's default group also requires an OMVS segment with a group ID.

BLZRACF and BLZRACFT contain similar RACF statements that you can use to create IDs. For the following sample RACF commands, replace the following placeholders with actual values: *#userid*, *#user-identifier*, *#group-name*, and *#group-identifier*.

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

Jazz RACF group creation for access to resources

During the server and Build System Toolkit installation and configuration, several directories are created to hold configuration and temporary files. These directories are created in the BLZCP* jobs that are shipped in the `hlq.SBLZSAMP` data set. The directories are identified in the various BLZCP* jobs as @confPath@ and @workPath@. By default, the directories are set to `/etc/jazz406` and `/u/jazz406`. Running these jobs creates the directories. The owner is the user ID who submits the jobs.

These jobs require the configuration of two RACF GROUPs that provide additional permission to other users who need access to the directories. If you are installing Jazz Team Server, see the sample RACF statements to perform this task in the relevant step in the BLZRACF job in `hlq.SBLZSAMP`. If you are installing the Build System Toolkit, the same RACF commands are provided in the BLZRACFT job in `hlq.SBLZSAMP`.

The following sample RACF commands create the JAZZCONF and JAZZWORK groups. Replace the *#conf-group-id* and *#work-group-id* placeholders with valid OMVS IDs.

Important: You must create these groups before you submit the BLZCP* jobs, or the jobs will fail.

```
ADDGROUP JAZZCONF OMVS(GID(#conf-group-id))
    DATA('GROUP WITH OMVS SEGMENT FOR JAZZ CONFIG DIRECTORIES')
ADDGROUP JAZZWORK OMVS(GID(#work-group-id))
    DATA('GROUP WITH OMVS SEGMENT FOR JAZZ CONFIG DIRECTORIES')
```

In general, you can control access to the USS configuration and work directories by limiting access to the directories that contain them. For example, access to `/etc/jazz406` can be restricted if the user or group does not have READ access to `/etc`.

Jazz Team Server security on z/OS systems

After you set up the Resource Access Control Facility (RACF) security options, you must complete the server installation and configuration on z/OS. To complete the server installation and configuration, you must have created JAZZCONF and JAZZWORK RACF GROUPs, as outlined in the general RACF considerations section, and completed the customization and submission of the BLZCP* jobs that are required for your configuration.

In addition, a few security considerations are specific to installing Jazz Team Server on z/OS. If you plan to run Jazz Team Server on z/OS, after you create the configuration and work directories by using the BLZCP* sample jobs, you must prepare the DB2 z/OS repositories by creating the databases, editing the `.properties` files, and running the appropriate repository tools functions. For more information, see Setting up a DB2 database on z/OS.

The basic requirements for creating the DB2 z/OS databases and running repository tools are as follows:

- A DB2 system administrator must create the databases for Jazz Team Server.
- A user ID and password must be created that have DBADM authority to the repositories and data warehouse. This user ID and password are used for all access to the DB2 z/OS repositories.
- In order to read and update the configuration files and logs, the user ID that runs the repository tools sample job to create the database tables, BLZCREDB, must be one of these IDs:
  - The same user ID that ran the BLZCP* sample configuration JCL
  - A member of the JAZZCONF and JAZZWORK RACF GROUPs
- Two additional user IDs are involved in populating and accessing the data warehouse:
  1. The first user ID is a data collection user ID, which must be a TSO ID with JazzAdmins access (READ access to the JazzAdmins EJBROLE profile). This user ID and password are specified during the setup process using the Jazz Team Server setup wizard.
  2. The second user ID is a report user who is granted SELECT access to the data warehouse tables as part of the data warehouse table creation process. By default, the user is RPTUSER. This user ID access can then be used if external products, such as , Rational Reporting for Development Intelligence, connect to the data warehouse.

WebSphere Application Server security setup

If you plan to run Jazz Team Server as a WebSphere Application Server application, you must set up and run several RACF profiles. Specifically, the user ID under which the application server runs must have READ and WRITE access to the CLM server configuration and work directories. Therefore, the user ID must be added to the JAZZCONF and JAZZWORK GROUPs.

In addition, each CLM user's repository permissions are determined by their permissions to specific RACF EJBROLE profiles. The EJBROLE profile definitions can be affected by whether an APPL profile was defined during the creation of the WebSphere Application Server profile itself. At least one user ID must be granted READ access to the JazzAdmins EJBROLE profile.

For additional details, see these topics:

- “Running Jazz Team Server and the applications for Rational solution for Collaborative Lifecycle Management applications on a z/OS system with WebSphere Application Server” on page 38
- “Setting up user security on a z/OS system by using RACF” on page 38

Several sections of BLZRACF address these requirements, including the definition of the EJBROLE profiles.

Tomcat server security setup

If you plan to run Jazz Team Server on z/OS with Tomcat, the user ID that is assigned to the Tomcat server job, either a normal user ID or a started task user ID, must be added to the RACF groups controlling access to the JAZZCONF and JAZZWORK directories. The Tomcat user ID must also have a valid OMVS segment.

The following sample RACF commands in BLZRACF connect the Tomcat user ID to the configuration and work RACF groups:

```
#  connect TOMCAT job or stc userid to JAZZCONF and JAZZWORK

   LISTGRP  JAZZCONF
   CONNECT  (TOMCATU) GROUP(JAZZCONF)
   LISTGRP  JAZZWORK
   CONNECT  (TOMCATU) GROUP(JAZZWORK)
```

Security for the Build System Toolkit and Rational Build Agent on z/OS systems

After you set up the Resource Access Control Facility (RACF) security options, you must complete the Build System Toolkit and Rational Build Agent configuration on z/OS. To complete the configuration, you must have created the JAZZCONF and JAZZWORK RACF GROUPs, as outlined in "RACF security on z/OS systems" on page 6, and completed the customization and submission of the BLZCP* jobs that are required for your configuration.

The next sections describe the additional security configurations that are specific to installing the Build System Toolkit and Rational Build Agent on z/OS.

Build System Toolkit RACF classes

Configuring the Build System Toolkit and Rational Build Agent depends on activating several RACF classes. The BLZRACFT sample member contains sample RACF statements to activate these classes.

- The STARTED CLASS assigns user ID relationships to the Rational Team Concert ISPF daemon started task and the Rational Build Agent started task.
- The APPL CLASS activates application protection for the ISPF daemon.
- The PTKTDATA CLASS supports PassTicket generation for the ISPF client.

Rational Team Concert ISPF daemon and client security setup

If you are installing and using the ISPF client, you must complete several RACF security steps. The basic tasks in the BLZRACFT sample job are as follows:

1. Create a group for the ISPF daemon started task user.
2. Create the ISPF daemon started task user (STCISPF).
3. Associate the ISPF-started tasks, BLZISPFS and BLZISPFD, with the STCISPF user ID.
4. Connect the STCISPF user ID to the groups that provide access to the configuration and work directories.
5. Allow STCISPF to run secure UNIX servers by granting access to the BPX.SERVER facility CLASS.
6. Allow STCISPF access to the PTKTDATA CLASS for PassTicket generation.

Important: If you use an ID other than STCISPF, change all references to that ID in BLZRACFT.

Program control

The Rational Team Concert ISPF daemon runs as a secure UNIX server. Servers with access to BPX.SERVER must run in a clean, program-controlled environment. Therefore, all programs that the ISPF client calls must also be program-controlled.

The Build System Toolkit components use the SYS1.LINKLIB library, the Language Environment run time (CEE.SCEERUN*) and the ISPF TSO/ISPF gateway (ISP.SISPLOAD) load library. Program control for ISP.ISPLOAD is configured when the TSO/ISPF gateway is configured.

For more information about the TSO/ISPF gateway, see the chapter *TSO/ISPF client gateway* in "ISPF Planning and Customizing" (GC34-4814)http://publibfp.dhe.ibm.com/epubs/pdf/ispzpc80.pdf.

The following sample RACF commands create the program control entries in the RACF database. For an example of RACF statements to perform this task, see the relevant step in the BLZRACFT job in `hlq.SBLZSAMP`, where `hlq` is the high-level qualifier that was specified during the SMP/E installation.

```
RALTER PROGRAM ** UACC(READ) ADDMEM(SYS1.LINKLIB//NOPADCHK)
RALTER PROGRAM ** UACC(READ) ADDMEM(CEE.SCEERUN//NOPADCHK)
RALTER PROGRAM ** UACC(READ) ADDMEM(CEE.SCEERUN2//NOPADCHK)
SETROPTS WHEN(PROGRAM) REFRESH
```

Note: Use the ** profile unless you already have a * profile in the PROGRAM class, in which case, do not use the ** profile because it obscures and complicates the search path that your security software uses. In this case, you must merge the existing * and the new ** definitions. For more details, see the "Security Server RACF Security Administrator's Guide" (SA22-7683) http://publibfp.dhe.ibm.com/epubs/pdf/ichza7b0.pdf.

Rational Build Agent security setup

You can run the Rational Build Agent in several ways. For details, see “Installing and configuring the Rational Build Agent on a z/OS system” on page 54. Run the Rational Build Agent under a user ID with UID(0) so that users can override the authority under which the build agent runs when they request a build. This user ID must be connected to the groups that allow access to the JAZZCONF and JAZZWORK directories. To run promotions, deployments, or other builds that use the ISPF gateway, this user ID must have READ access to the required ISPF configuration files (`ISPZXENV` and `ISPF.conf`), must be authorized to use TSO, and must have an ALIAS for a valid HLQ.

If you plan to run the Rational Build Agent as a started task, you must issue RACF commands to make the definitions to set up the started task. For an example of the RACF statements to perform this task, see the instructions in the BLZRACFT job in `hlq.SBLZSAMP`.

The following sample RACF commands create the BLZBFA started task, with protected user ID (STCBFA) and group STCGROUP assigned to them. Replace the `#group-id and #user-id-*` placeholders with valid OMVS IDs.

Note: Ensure that the started task user ID is protected by specifying the NOPASSWORD keyword.

```
 ADDGROUP STCGROUP OMVS(GID(#group-id)) DATA('GROUP WITH OMVS SEGMENT FOR STARTED TASKS')

 ADDUSER STCBFA DFLTGROUP(STCGROUP) NOPASSWORD NAME('RATIONAL BUILD AGENT') OMVS(UID(0)
HOME(/tmp) PROGRAM(/bin/sh)) DATA('RATIONAL TEAM CONCERT')

 RDEFINE STARTED BLZBFA.* DATA('RTC - RATIONAL BUILD AGENT') STDATA(USER(STCBFA)
 GROUP(STCGROUP) TRUSTED(NO))

 SETROPTS RACLIST(STARTED) REFRESH
```

```
#  connect Build Forge Agent userid to JAZZ config group (default JAZZCONF)

   LISTGRP  JAZZCONF
   CONNECT (STCBFA) GROUP(JAZZCONF)

#  connect Build Forge Agent userid to JAZZ work group (default JAZZWORK)

   LISTGRP  JAZZWORK
   CONNECT (STCBFA) GROUP(JAZZWORK)
```

User ID OMVS segment creation

For each ISPF client user and for each Rational Developer for System z user, you must define a RACF OMVS segment or equivalent that specifies a valid non-zero z/OS UNIX user ID (UID), home directory, and shell command. In addition, if you run builds through the Rational Build Agent and override the user authentication, and you set the "load directory" in the build to your OMVS home directory, you must have an OMVS segment for user who submit personal dependency builds. The users default group also requires an OMVS segment with a group ID.

In the following sample RACF commands, replace the following placeholders with actual values: `#userid`, `#user-identifier`, `#group-name`, and `#group-identifier`.

```
ALTUSER #userid
OMVS(UID(#user-identifier) HOME(/u/#userid) PROGRAM(/bin/sh) NOAS-SIZEMAX)
ALTGROUP #group-name OMVS(GID(#group-identifier))
```

Although it is not recommended, you can use the shared OMVS segment defined in the BPX.DEFAULT.USER profile of the FACILITY class to fulfill the OMVS segment requirement for Rational Team Concert.

Additional access to configuration and work directories

The following additional users need access to the work directories. The users must be connected to the JAZZWORK group.
- ISPF client users
- Enterprise Extensions build users

Rational Team Concert Job Monitor security setup

If you plan to run JCL-based builds through the Rational Build Agent, you must consider additional security tasks when you configure the Job Monitor. The greatest flexibility for users to submit JCL under their own IDs is provided if the build agent that supports these builds is started by user ID UID(0).

Rational Developer for System z integration feature security setup

If you plan to run the Rational Developer for System z integration feature with Rational Team Concert, the user ID that is assigned to the Remote System Explorer daemon (RSED) started task must be added to the RACF groups that control access to the JAZZCONF and JAZZWORK directories. To be able to store SCM metadata in the working directories, users of the integration feature must be connected to the JAZZWORK group.

The following sample RACF commands connect the RSED started task user ID to the configuration and work RACF groups:

```
#  connect RSED Started task userid to JAZZCONF and JAZZWORK

   LISTGRP  JAZZCONF
   CONNECT  (RSED) GROUP(JAZZCONF)
   LISTGRP  JAZZWORK
   CONNECT  (RSED) GROUP(JAZZWORK)
```

Chapter 2. Setting up the server on z/OS

Installing and configuring Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications on z/OS systems

Installing the Jazz Team Server and other Rational solution for Collaborative Lifecycle Management (CLM) products on z/OS systems is completed using SMP/E.

The SMP/E package contains several FMIDs. You can choose to install only the FMIDs that you require based on your server configuration. Use the instructions in:

If you are not installing the Jazz Team Server on your z/OS system, skip to the next chapter.

Configuring your installation

Before you can configure the Jazz Team Server and the Rational solution for Collaborative Lifecycle Management (CLM) products on z/OS, you must ensure that the SMP/E installation has ended successfully.

About this task

If you are using the 64-bit version of Java™ 6 or any supported Java version, you must increase the settings that affect the amount of shared memory pages available to the JVM. You can find the z/OS UNIX System Services system parameters using the **`D OMVS,L`** command. The parameters that are affected are:

- **`MAXMMAPAREA`** must be greater than 250,000 for a single server
- **`MAXSHAREPAGES`** must be greater than 250,000 for a single server

Note: If you run more than one server simultaneously, multiply the value you set for these parameters by the number of servers that you run.

Creating directories for Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications

The Jazz Team Server (JTS) and each of the Rational solution for Collaborative Lifecycle Management (CLM) applications installed on z/OS require several directories in addition to the UNIX System Services directory created by the SMP/E installation.

One of the required directories is a working directory and the other required directories are used to store Jazz Team Server and CLM configuration files. There are sample members for each application in *`hlq`*`.SBLZSAMP` used to create and populate these directories with the required files from the SMP/E installed directory. The following table lists the sample member in *`hlq`*`.SBLZSAMP` associated with the installed application.

Table 1. Sample member names for each application installed on z/OS.

Installed application	Sample member
Jazz Team Server (JTS) - HRJS406	`BLZCPJTS`

Table 1. Sample member names for each application installed on z/OS. (continued)

Installed application	Sample member
Change and Configuration Management (CCM) - HRCM406	`BLZCPCCM`
Quality Management (QM) - HRQM406	`BLZCPQM`
Requirements Management (RM) - HRRM406	`BLZCPRM`
Apache Tomcat - HROC406	`BLZCPOPT`

You can run sample JCL members based on the components that you installed and plan to configure.

There are also sample members used for configuring the Build System Toolkit (`BLZCPBTK`) and the Rational Build Agent (`BLZCPBFA`) on z/OS. For instructions on running those members, see:
- "Creating additional Build System Toolkit directories" on page 45
- "Creating additional Rational Build Agent directories" on page 54

The following three symbolic names are used throughout this guide to indicate the following directories:

Table 2. Symbolic names for directories

Symbolic name	Use	Variable in BLZCP* jobs	Default directory
@pathPrefix@	The path prefix specified during the SMP/E installation.	BLZHOME	
@confPath@	The Jazz Team Server configuration files directory.	BLZCONF	`/etc/jazz406`
@workPath@	The Jazz Team Server working directory.	BLZWORK	`/u/jazz406`

You must use unique directory names for @confPath@ and @workPath@ if you have additional version 2, version 3.x, or version 4.0.x installations.

Space recommendations:
The JTS and CLM directories must have the following allocations:

@workPath@
3000 cylinders 3390, for running the Jazz Team Server and CLM applications on z/OS. This figure might be larger than required if the server is not running on z/OS. Allocate a file system that allows for significant expansion as your installation grows.

@confPath@
Five cylinders 3390

RACF server file access requirements: The BLZCP* jobs will create directories that are owned by the user ID that runs the BLZCP* jobs. In addition, the BLZCP* members contain the @confgrp@ and @workgrp@ variables, which must be set to the SAF groups that contain all of the user IDs that require write access to the configuration and work directories. You must also associate group IDs (GID) with these SAF groups. The Jazz Team Server and CLM application directories require the following RACF file access permissions so the server can function:

1. @confPath@ and @workPath@ must be accessible for read and write by the user ID under which the Tomcat server or WebSphere Application Server runs.
2. @confPath@ and @workPath@ must be accessible for read and write by the user ID under which the Jazz Team Server repository tools are run.

The recommended approach to providing access to the working directories and configuration directories is to create a RACF group and assign the IDs to that group that are needed for the functions you plan to use.

In addition to creating and populating these directories, the sample jobs provided also perform several required customizations. The variables mentioned in the Symbolic name table, must be set in several places in each job. In addition there are other variables that might need to be set listed in the following table:

Table 3. Additional variable names

Variable Name	Use	Default value
BLZBFAH	The installation directory of the Rational Build Agent.	`/usr/lpp/jazz/v4.0.6/buildagent`
BLZBFAC	The Rational Build agent configuration files directory.	`/etc/jazz406/ccm`
BLZJAVA	The location of the Java directory.	`/usr/lpp/java/J6.0`
BLZHOST	The fully qualified host name where the server is running .	`Host.name`
iconvLoc	The location of the iconv utility.	`/bin/iconv`

The instructions contained in each configuration job will indicate which variables you must configure.

For each JOB, configure the sample member in *hlq*.SBLZSAMP using the instructions contained in the member. Use the **SUBMIT** command to submit the modified JCL and check the job log. Return codes of 0 indicate the configuration is correct.

Customizing the configuration files for z/OS systems

You must modify several Jazz Team Server (JTS) and Rational solution for Collaborative Lifecycle Management (CLM) configuration files for z/OS based on the directories you selected for `@pathPrefix@`, `@workPath@`, and `@confPath@`. Most of these modifications are performed automatically by the `BLZCP*` jobs you configured and submitted previously. However, if you need to make additional modifications, these instructions explain how to modify the configuration files.

The Jazz Team Server properties files for z/OS are ASCII files. You can use one of the following techniques to edit ASCII files under z/OS UNIX:

- If you have z/OS 1.10 or later or z/OS 1.9 with the PTF for APAR OA22250, use ISPF option 3.17 to edit ASCII files under z/OS UNIX System Services.
- If you use Rational Developer for System z, use Remote System Explorer to connect and modify the files.
- Download or use FTP to transfer the files to a Windows PC, modify them, and transfer them back to the z/OS system.
- If you have other tools for editing ASCII files under z/OS UNIX System Services, use those tools.

Customizing the provisioning profiles:

This topic describes the changes required to customize the provisioning profiles.

About this task

Note: These instructions are provided in case you need to customize the provisioning files with additional modifications. Typically modifications are performed automatically by the `BLZCP*` jobs.

If you used the `@pathPrefix@` setting during SMP/E installation, edit the files listed below for the applications that you have installed to ensure the file URL points to the absolute path of the application update sites. Modifications are only required for the Rational solution for Collaborative Lifecycle Management (CLM) applications that are installed.

Jazz Team Server (JTS)

```
@confPath@/jts/provision_profiles/nl1_profile.ini
@confPath@/jts/provision_profiles/profile.ini
@confPath@/jts/provision_profiles/nl2_profile.ini
@confPath@/jts/provision_profiles/license-profile.ini
@confPath@/jts/provision_profiles/clm-activation.ini
```

Change and Configuration Management (CCM)

```
@confPath@/ccm/provision_profiles/nl2_rtc-commons-profile.ini
@confPath@/ccm/provision_profiles/nl1_rtc-commons-profile.ini
@confPath@/ccm/provision_profiles/nl2_enterprise-profile.ini
@confPath@/ccm/provision_profiles/nl1_enterprise-profile.ini
@confPath@/ccm/provision_profiles/enterprise-update-site.ini
@confPath@/ccm/provision_profiles/nl1_profile.ini
@confPath@/ccm/provision_profiles/rtc-commons-profile.ini
@confPath@/ccm/provision_profiles/nl2_profile.ini
@confPath@/ccm/provision_profiles/profile.ini
```

Quality Management (QM)

```
@confPath@/qm/provision_profiles/profile.ini
@confPath@/qm/provision_profiles/rqm-nl1-profile.ini
@confPath@/qm/provision_profiles/nl2_profile.ini
@confPath@/qm/provision_profiles/rqm-prereqs-profile.ini
@confPath@/qm/provision_profiles/rqm-profile.ini
@confPath@/qm/provision_profiles/rqm-nl2-profile.ini
@confPath@/qm/provision_profiles/nl2_rtc-commons-profile.ini
@confPath@/qm/provision_profiles/nl1_rtc-commons-profile.ini
@confPath@/qm/provision_profiles/nl1_profile.ini
```

Requirements Management (RM)

```
@confPath@/rm/provision_profiles/fronting-server-profile.ini
@confPath@/rm/provision_profiles/rrcweb-profile.ini
@confPath@/rm/provision_profiles/nl2-fronting-server-profile.ini
@confPath@/rm/provision_profiles/rrcweb-rrcx-profile.ini
@confPath@/rm/provision_profiles/nl2-fronting-server-web-profile.ini
@confPath@/rm/provision_profiles/nl1-fronting-server-web-profile.ini
@confPath@/rm/provision_profiles/nl2-rrcweb-profile.ini
@confPath@/rm/provision_profiles/nl2-rrcweb-rrcx-profile.ini
@confPath@/rm/provision_profiles/fronting-server-web-profile.ini
@confPath@/rm/provision_profiles/nl1-fronting-server-profile.ini
@confPath@/rm/provision_profiles/nl1-rrcweb-rrcx-profile.ini
@confPath@/rm/provision_profiles/nl1-rrcweb-profile.ini
```

For example, if `@pathPrefix@` is set to *myroot*, then this setting resolves to `url=file:/`**`myroot`**`/usr/lpp/jazz/v4.0.6/server/conf/ccm/sites/update-site`.

Customizing the logging utility files:

The `log4j.properties` file configures the logging framework used by the Jazz Team Server and the other Rational solution for Collaborative Lifecycle Management (CLM) applications on z/OS systems.

There is a `log4j.properties` file included in each of the following application configuration directories, for each of the applications that you have installed and configured:

- `@confPath@/jts/log4j.properties`
- `@confPath@/ccm/log4j.properties`
- `@confPath@/qm/log4j.properties`
- `@confPath@/rm/log4j.properties`
- `@confPath@/admin/log4j.properties`

By default, the `log4j.properties` file can write Jazz Team Server and CLM application messages to the WebSphere Application Server servant JOB log or the Tomcat server JOB log, and to any additional files that are specified in the `log4j.properties` file. Use these additional log files in conjunction with the server job logs for problem determination.

The `log4j.properties` file is updated automatically by the `BLZCP*` jobs, but if you need to make additional changes, modify the line that begins with `log4j.appender.file.File=` so that it includes the value you are using for your Jazz Team Server and CLM applications `@workPath@` directories. For example:

```
log4j.appender.file.File=/u/jazz406/ccm/logs/ccm.log
```

Setting up a DB2 database on a z/OS system

This section provides information about setting up a DB2 database on a z/OS system.

Before you begin

If you use a Derby database on z/OS instead of DB2, you can skip to "Running Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications on z/OS with Tomcat and Derby" on page 36.

About this task

These instructions describe how to connect to DB2 on z/OS from a server on z/OS or from a server on an operating system other than z/OS. The DB2 setup tasks should be performed by your DB2 administrator.

Prerequisites to set up a DB2 database on a z/OS system

Ensure that DB2 Version 9.1 or Version 10.1 for z/OS with the Universal JDBC driver is installed and running on the z/OS system that is used as the database server. You must use a DB2 mode that supports the new functions introduced in version 9.1. These prerequisites are required if the server is on z/OS or is on another operating system and connecting to DB2 for z/OS.

On z/OS, the Jazz Team Server requires the WLM procedures associated with the DB2 stored procedures SYSPROC.DSNUTILS and SYSIBM.SQLxxx to be active. DB2 WLM-managed stored procedures also require z/OS Resource Recovery Services (RRS) to be active. If necessary, verify that the stored procedures are active by comparing the names of the DB2 WLM environment variables with the active WLM procedures. Use the SQL **SELECT** command to retrieve the WLM procedure names through DB2 SPUFI or your preferred technique:

```
SELECT DISTINCT WLM_ENVIRONMENT FROM SYSIBM.SYSROUTINES WHERE
  (NAME='DSNUTILS' OR (SCHEMA='SYSIBM' AND NAME LIKE 'SQL%'));
```

This command produces results like the following example:

```
---------+---------+---------+---------+---------+---------+---
WLM_ENVIRONMENT
---------+---------+---------+---------+---------+---------+---
DSN9WLM1
```

Use the following command from the z/OS console to display the WLM active procedures: **D WLM,APPLENV=***. This command produces results like the following:

```
SDSF SYSLOG   2493.101 2094 2094 07/10/2008 0W   14886  COMMAND ISSUED
 RESPONSE=RALNS32
  IWM029I  16.31.12  WLM DISPLAY 465
    APPLICATION ENVIRONMENT NAME     STATE     STATE DATA
    BBOASR1                          AVAILABLE
    BBOASR2                          AVAILABLE
    BBOC001                          AVAILABLE
    CBINTFRP                         AVAILABLE
    CBNAMING                         AVAILABLE
    CBSYSMGT                         AVAILABLE
    DSN9WLM1                         AVAILABLE
    DSN9WLM2                         AVAILABLE
    DSN9WLM3                         AVAILABLE
```

In this case, you can see that the DSN9WLM1 procedure is active.

Setting up DB2 for z/OS to use with Jazz Team Server

When running the Jazz Team Server and the Rational solution for Collaborative Lifecycle Management (CLM) applications with DB2 for z/OS, you must create a DB2 storage group and several DB2 databases, depending on which CLM applications you plan to use. You must also authorize a user to the storage group and databases.

About this task

The following steps must be performed before running the repository tools database builder utility, which creates the repository tables in each database. None of these steps is performed by the server database builder utility.

Creating a storage group

The storage group must be appropriate to the system. The following example shows a DB2 SQL create statement:

```
CREATE STOGROUP CLMSTG VOLUMES ('*') VCAT yourHlq ;
```

Notes:

1. The storage group can be named something other than *CLMSTG*.
2. *yourHlq* is the high-level qualifier of your DB2 files. It must exist on your system, and the Jazz Team Server user must have full access to it.

Creating databases

The databases must be created with *UNICODE* as the CCSID. Multiple databases are required to support the Jazz Team Server and the other CLM applications. If you do not plan to use a particular application, you do not need to create that database. The following example shows DB2 SQL create statements:

```
Create the following Database for the Jazz Team Server
CREATE DATABASE JTS406 STOGROUP CLMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the CCM application
CREATE DATABASE CCM406  STOGROUP CLMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the QM application
CREATE DATABASE QM406  STOGROUP CLMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the Data Warehouse
CREATE DATABASE DW406  STOGROUP CLMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;
```

Notes:

1. You can replace the database names on the `CREATE DATABASE` statement with a different name.
2. The database name is used later for the `teamserver.properties` `com.ibm.team.repository.db.db2.dsn.dbname` property settings.
3. *BP16K0* is an example of the buffer pool name. (On z/OS, a 16K page size or larger is required.) This buffer pool is used for creating tables. Table spaces are created in the default 16K buffer pool, unless you selected a larger buffer pool.
4. You must create your DB2 database with *UNICODE* as the CCSID, otherwise the create database task fails and this message is displayed: `CRJAZ0249I The database code page was set to "E" but should be "U". Recreate the database with the correct code page.`

5. You can define these databases in a single DB2 subsystem; however, you must also specify unique values for each `teamserver.properties` file directive `com.ibm.team.repository.db.schemaPrefix` to separate Jazz repositories as described in "Customizing the Jazz Team Server and Rational solution for Collaborative Lifecycle Management properties files for DB2 on z/OS" on page 23.
6. The reporting function in CLM requires a data warehouse to operate. You should also add the property `com.ibm.team.datawarehouse.db.schemaPrefix` for the data warehouse tables if you plan to implement the data warehouse.

Authorizing user access to the databases

The server requires a user ID and password to access the repositories. The user ID and password are specified later in the **`teamserver.properties`** file. This user ID is not used to log on to the server. It is used only to provide authority for the server to access the DB2 for z/OS databases. Specifically, this user ID requires permissions as shown in the example. In this example, the user has the name *jazz*.

A user ID with SYSADM access to the databases has the appropriate access to run the server and create the required tables and indexes using the repository tools **`-createTables`** command.

If your process does not allow a user ID with SYSADM access, you must grant additional permissions to the user ID. Examples of the GRANT statements are included in the following sample. (Comments are indicated by `--`.)

```
-- General
GRANT USE OF STOGROUP CLMSTG TO jazz ;
GRANT SELECT ON SYSIBM.SYSTABLES TO jazz ;
GRANT SELECT ON SYSIBM.SYSINDEXES TO jazz ;
GRANT SELECT ON SYSIBM.SYSDATABASE TO jazz ;
GRANT SELECT ON SYSIBM.SYSTABCONST TO jazz ;
GRANT SELECT ON SYSIBM.SYSAUXRELS TO jazz ;
GRANT SELECT on SYSIBM.SYSKEYS TO jazz ;

-- Grant access to bufferpool.  Default bufferpool is used
-- for tablespaces and an additional grant for the default BP
-- may be needed if different from this one
GRANT USE OF BUFFERPOOL BP16K0 TO jazz ;

-- JTS – if the JTS repository will be on DB2 z/OS
GRANT DBADM ON DATABASE JTS406 TO jazz ;

-- CCM – if the CCM repository will be on DB2 z/OS
GRANT DBADM ON DATABASE CCM406 TO jazz ;

-- QM – if the QM repository will be on DB2 z/OS
GRANT DBADM ON DATABASE QM406 TO jazz ;

-- If you plan to use Data Warehouse reporting where "DWX"
--  equals the prefix you plan to use for the DW schemas
GRANT DBADM ON DATABASE DW406 TO jazz ;

GRANT CREATEIN ON SCHEMA DWX_CFG TO USER99;
GRANT CREATEIN ON SCHEMA DWX_ODS TO USER99;
GRANT CREATEIN ON SCHEMA DWX_ASSET TO USER99;
GRANT CREATEIN ON SCHEMA DWX_SCHK TO USER99;
GRANT CREATEIN ON SCHEMA DWX_DW TO USER99;
GRANT CREATEIN ON SCHEMA DWX_CALM TO USER99;

COMMIT ;
```

If you used an ID with DBADM and these additional privileges, you must also grant access to the Views after the tables are created. For details, see "Creating database tables using repository tools" on page 26.

In addition, if the value of field **DBADM CREATE AUTH** is set to **NO** on panel **DSNTIPP** during DB2 installation, you must grant **SYSADM** authorization to the user.

```
GRANT SYSADM TO jazz ;
COMMIT ;
```

If the value of field **DBADM CREATE AUTH** is set to **YES** on panel **DSNTIPP** during DB2 installation, the user can create the database with **DBADM** authority, but if you want the user to upgrade the database, you must grant **SYSCTRL** authorization to the user.

```
GRANT SYSCTRL TO jazz ;
COMMIT ;
```

In addition, as part of the data warehouse creation and server setup process, a report user is defined for the data warehouse. This user ID is automatically granted "connect" and "select" access to the data warehouse tables as part of the configuration process.

Customizing the Jazz Team Server and Rational solution for Collaborative Lifecycle Management properties files for DB2 on z/OS

This section describes how to edit and configure the `teamserver.properties` files so that the server on z/OS can connect to DB2. The Jazz Team Server (JTS) and the other Rational solution for Collaborative Lifecycle Management (CLM) applications each have `teamserver.properties` files that contain server properties.

About this task

Locate the configuration directories for each application:

- Jazz Team Server (JTS) - *@confPath@*`/jts`
- Change and Configuration Management (CCM) - *@confPath@*`/ccm`
- Quality Management (QM) - *@confPath@*`/qm`
- Requirements Management (RM) - uses the database connection information in the Jazz Team Server `teamserver.properties`, so you do not have to repeat this step for the RM application.

On z/OS, the *@confPath@* is the path value that was used when customizing the server using the BLZCP* sample jobs, such as `/etc/jazz406`. On systems other than z/OS, *@confPath@* is the installation path for the configuration directory, such as `/opt/IBM/JazzTeamServer/server/conf` on Linux.

By default, the `teamserver.properties` files in those directories are configured to use a Derby repository and data warehouse for reporting. Edit each `teamserver.properties` file to use DB2 for z/OS according to the instructions in this topic.

Use the **-DIS DDF** command for your DB2 for z/OS subsystem to display some of the values you need to supply. For example, you can retrieve the *location*, *ipaddr*, and *port* (*tcpport*) from the following display:

```
-DSN9 DIS DDF
 DSNL080I  -DSN9 DSNLTDDF DISPLAY DDF REPORT FOLLOWS: 548
 DSNL081I STATUS=STARTD
 DSNL082I LOCATION           LUNAME            GENERICLU
 DSNL083I NS32DB             NETA.NS32DB       -NONE
 DSNL084I TCPPORT=3500  SECPORT=3510  RESPORT=3501  IPNAME=-NONE
 DSNL085I IPADDR=9.42.81.74
 DSNL086I SQL    DOMAIN=RALNS32.example.com
 DSNL086I RESYNC DOMAIN=RALNS32.example.com
 DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Modify each teamserver.properties file to match the database configuration you created in previous steps, and also to match your DB2 z/OS configuration. Edit the following properties according to your configuration:

- *ipAddress,*
- *ipPort*
- *location*
- *user*
- *password*
- *dbname*
- *schemaprefix*

Notes:

- There can be up to four repositories: one repository for JTS, one repository for each installed component that has a database (CCM and QM), and one repository for the data warehouse.
- Each of these repositories must have a unique database name and schema prefix (if they are in a single DB2 z/OS subsystem).
- Each properties file will point to the same data warehouse configuration.

To configure the server or application to use DB2 for z/OS, edit each of the teamserver.properties files in the configuration directories for the Jazz Team Server or other installed CLM application. For example, *@confPath@*/jts/teamserver.properties and *@confPath@*/ccm/teamserver.properties files and locate the following lines. This example is for CCM:

```
# JDBC Repository DB location, specifying this property disables
# system-based selection of default location
# NOTE THAT EVERY APPLICATION INSTANCE AND JAZZ TEAM SERVER
# REQUIRES ITS OWN UNIQUE DATABASE
com.ibm.team.repository.db.vendor = DERBY
com.ibm.team.repository.db.jdbc.location=conf/ccm/derby/repositoryDB

# JDBC Data Warehouse DB location, specifying this property disables
# system-based selection of default location
com.ibm.team.datawarehouse.db.vendor=derby_net
com.ibm.team.datawarehouse.db.jdbc.location=//localhost:1527/conf/jts
                       /derby/warehouseDB
```

Add a # in column one to comment out the following properties for both the com.ibm.team.repository and com.ibm.team.datawarehouse properties like this:

```
# JDBC Repository DB location, specifying this property disables
# system-based selection of default location
# NOTE THAT EVERY APPLICATION INSTANCE AND JAZZ TEAM SERVER REQUIRES ITS OWN
# UNIQUE DATABASE
```

```
#com.ibm.team.repository.db.vendor = DERBY
#com.ibm.team.repository.db.jdbc.location=conf/ccm/derby/repositoryDB

# JDBC Data Warehouse DB location, specifying this property disables
# system-based selection of default location
#com.ibm.team.datawarehouse.db.vendor=derby_net
#com.ibm.team.datawarehouse.db.jdbc.location=//localhost:1527/conf/jts
                       /derby/warehouseDB
```

Uncomment the lines shown in the examples and edit to match the database configuration you created in previous steps, and also to match your DB2 configuration.

Edit the *location*, *user*, *password*, and *dbname* properties according to your configuration.

Specifically, edit the following lines in each `teamserver.properties` file for that repository:

1. Ensure this line is uncommented:

   ```
   com.ibm.team.repository.db.vendor = db2z
   ```
2. In the following line:

   ```
   com.ibm.team.repository.db.jdbc.location=//ipAddress:ipPort/
   location:user=jazzDBuser;password={password};
   ```

   replace:
   - *ipAddress* with your ipaddr.
   - *ipPort* with your tcpport.
   - *location* with the value listed in the DDF report under LOCATION.
   - *jazzDBuser* with the user ID you created, which has appropriate access to the DB2 database.

     Note: Do not modify *password*={password}.
3. In the following line:

   ```
   com.ibm.team.repository.db.jdbc.password=jazzDBpswd
   ```

   replace *jazzDBpswd* with the password for your DB2 z/OS user.
4. In the following line:

   ```
   com.ibm.team.repository.db.db2.dsn.dbname=JAZZDB
   ```

   replace *JAZZDB* with the name of the database you created for this component.
5. In the following line:

   ```
   #com.ibm.team.repository.db.schemaPrefix=xx
   ```

   you must remove the # and replace *xx* with a **unique** prefix for each database in the same DB2 z/OS subsystem.

   Note: To create several Jazz databases in the same DB2 subsystem, you must differentiate the table owners for the Jazz tables. To do so, the Jazz Team Server uses the `com.ibm.team.repository.db.schemaPrefix` directive to add a prefix to the Jazz DB2 objects so that they are unique within a DB2 subsystem. The prefix set in `com.ibm.team.repository.db.schemaPrefix` is added to the owner prefix along with an underscore. For example, the `CREATOR` will be modified to `JTS406_REPOSITORY`, in a given database when `com.ibm.team.repository.db.schemaPrefix=`**`JTS406`**. Ensure a unique reference using `CREATOR` and `TABLE` name.

Update the `teamserver.properties` files for references to the data warehouse on DB2 z/OS.

Specifically, edit the following lines:

1. Comment out the following properties related to Derby data warehouse configuration (the third property might not be listed):

   ```
   # com.ibm.team.datawarehouse.db.vendor=DERBY
   # com.ibm.team.datawarehouse.db.jdbc.location=conf/jts/derby/warehouseDB
   # com.ibm.team.datawarehouse.db.net.port=1527
   ```

2. Uncomment the following properties and make similar updates as for the repository.

   ```
   #com.ibm.team.datawarehouse.db.vendor = db2z
   #com.ibm.team.datawarehouse.db.jdbc.location=//ipAddress:ipPort/
   #location:user=jazzDBuser;password={password};
   #com.ibm.team.datawarehouse.db.jdbc.password=jazzDBpswd
   # The database user for whom proper permissions will be granted
   #com.ibm.team.datawarehouse.report.user = RPTUSER
   #com.ibm.team.datawarehouse.db.db2.dsn.dbname=RIDW
   #com.ibm.team.datawarehouse.db.schemaPrefix=DW406
   ```

   You must add the following property to specify a prefix of your choice for the data warehouse tables:

   ```
   com.ibm.team.datawarehouse.db.schemaPrefix=DW406
   ```

3. In `com.ibm.team.datawarehouse.report.user = RPTUSER`, enter the report user for whom the proper permission will be granted. The default is `RPTUSER`.

   Note: If you use the setup wizard, proper permission will be granted to the report user. If you want to manually grant permissions, this user must be able to do SELECT on the database to view a report.

   Specifically, when you run the repository tools **`–createWarehouse`** or initialize the data warehouse using the setup wizard, `RPTUSER` is being granted permission to select from the tables, using statements such as:

   ```
   GRANT SELECT ON RIDW.VW_RQST_HISTORY TO $REP_USER;
   ```

Finally, replace all instances of *@workPath@* with the path that you selected for *@workPath@* if they are not already modified. The *@workPath@* replacement should have been done when you ran the `BLZCP*` jobs. For example, these properties:

- `com.ibm.team.fulltext.indexLocation=@workPath@/workitemindex`
- `com.ibm.team.repository.tmpdir=@workPath@/contentservice`
- `com.ibm.team.scm.tmpdir=@workPath@/contentservice`
- `com.ibm.team.scm.vcs.tmpdir=@workPath@/versionedcontentservice`

Tip: The Jazz database contains LOB (large object) columns in its tables. The LOB columns are associated with a buffer pool that is created by DB2 during table creation. The default buffer pool for a LOB tablespace is defined in zparm **`TBSBPLOB`**. You can set the value to a 16K buffer pool if you do not want 8K page sizes. For additional information see "Alternatives in defining LOBs" in the IBM Redbook *LOBs with DB2 for z/OS: Stronger and Faster* at http://www.redbooks.ibm.com/redbooks/pdfs/sg247270.pdf.

Creating database tables using repository tools

When you use DB2 for z/OS, initially you must use the repository tools to create the required database tables and indexes for the Jazz Team Server, any Rational solution for Collaborative Lifecycle Management (CLM) applications you plan to use, and the data warehouse.

About this task

Ensure the `teamserver.properties` file has been configured correctly as described in "Customizing the Jazz Team Server and Rational solution for Collaborative Lifecycle Management properties files for DB2 on z/OS" on page 23 before you create the database tables. If the server is installed on z/OS and will run on z/OS, you can use sample JCL provided to create the repository tables.

Procedure

1. Configure member `BLZCREDB` in *`hlq`*`.SBLZSAMP` by following the instructions in the sample JCL. Copy the member to another member before you customize it.
   - The Jazz Team Server and the repository tools require access to the DB2 V9 or V10 JDBC license and JDBC .jar files on your system. These files are named `db2jcc_license_cisuz.jar` and `db2jcc.jar` and are located by default at `/usr/lpp/db2910_jdbc/classes` for V9 and `/usr/lpp/db2a10/jdbc/classes` for V10.
   - If the file is in a different directory, then modify the sample job with the correct location.
2. Submit the modified JCL and check the job log. The following message must end the STDOUT: `The database tables were created successfully.` The details are in the JCL sample.

   Note: You must run this job several times. Run it to create each of the following tables:
   - JTS tables using `application = jts` and `repotoolsCommand = -createTables`
   - Data warehouse tables using `application = jts` and `repotoolsCommand = -createWarehouse`
   - CCM tables using `application = ccm` and `repotoolsCommand = -createTables`
   - QM tables (if required) using `application = qm` and `repotoolsCommand = -createTables`

   Pay particular attention to the variable in the job `@appl@`. Change all instances of `@appl@` to the application tables you are creating.

   You can adapt BLZCREDB to run other repository tools functions by making a copy of the job and modifying the `//MAINARGS DD *` section of the job to reflect the repository tools commands you want to run. You can also use member BLZRPOTL as a template for running other repository tools commands.

What to do next

After you create the database tables, if you used a user ID with DBADM authority instead of a user ID with SYSADM authority, you must grant additional permissions to the views created by the table creation process. Use GRANT statements like the following examples. The GRANT statements must include the prefix you selected for your repositories. (Comments are indicated by `--`.)

```
-- Grants for JTS assuming schema prefix of JTSX and user ID of jazz

GRANT DELETE, INSERT, SELECT, UPDATE
    ON TABLE
        JTSX_COMMON_SNAPSHOT.SMPL_ITER,
        JTSX_REPOSITORY.CONTRIBUTOR,
        JTSX_REPOSITORY_SNAPSHOT.LATEST
      TO JAZZ ;
```

```
-- Grants for CCM assuming schema prefix of CCMX and user ID of jazz

GRANT DELETE, INSERT, SELECT, UPDATE
      ON TABLE
         CCMX_BUILD_SNAPSHOT.JUNIT_CNT,
         CCMX_COMMON_SNAPSHOT.SMPL_ITER,
         CCMX_REPOSITORY.CONTRIBUTOR,
         CCMX_REPOSITORY_SNAPSHOT.LATEST,
         CCMX_WORKITEMS_SNAPSHOT.NEW_WI_COUNT,
         CCMX_WORKITEMS_SNAPSHOT.WC_BIGDECIMAL_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_BIGDECIMAL_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_BOOL_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_BOOL_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_DOUBLE_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_DOUBLE_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_FLOAT_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_FLOAT_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_INT_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_INT_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_LARGE_STRING_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_LARGE_STRING_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_LONG_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_LONG_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_MEDIUM_STRING_EXT_FROM
      TO JAZZ ;

GRANT DELETE, INSERT, SELECT, UPDATE
      ON TABLE
         CCMX_WORKITEMS_SNAPSHOT.WC_MEDIUM_STRING_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_STRING_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_STRING_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WC_TIMESTAMP_EXT_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WC_TIMESTAMP_EXT_TO,
         CCMX_WORKITEMS_SNAPSHOT.WORKITEM_CHNGS_FROM,
         CCMX_WORKITEMS_SNAPSHOT.WORKITEM_CHNGS_TO,
         CCMX_WORKITEMS_SNAPSHOT.WS_BIGDECIMAL_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_BOOL_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_DOUBLE_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_FLOAT_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_INT_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_LARGE_STRING_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_LONG_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_MEDIUM_STRING_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_STRING_EXT,
         CCMX_WORKITEMS_SNAPSHOT.WS_TIMESTAMP_EXT
      TO JAZZ ;

-- Grants for QM assuming schema prefix of QMX and user ID of jazz

GRANT DELETE, INSERT, SELECT, UPDATE
      ON TABLE
         QMX_COMMON_SNAPSHOT.SMPL_ITER,
         QMX_PLANNING_SNAPSHOT.DEFECT_FACT,
         QMX_REPOSITORY.CONTRIBUTOR,
         QMX_REPOSITORY_SNAPSHOT.LATEST,
         QMX_RESERVATION_SNAPSHOT.GROUP_RESERVATION_DAY_VIEW,
         QMX_RESERVATION_SNAPSHOT.RESERVATION_DAY_VIEW,
         QMX_WORKITEMS_SNAPSHOT.NEW_WI_COUNT,
         QMX_WORKITEMS_SNAPSHOT.WC_BIGDECIMAL_EXT_FROM,
         QMX_WORKITEMS_SNAPSHOT.WC_BIGDECIMAL_EXT_TO,
         QMX_WORKITEMS_SNAPSHOT.WC_BOOL_EXT_FROM,
         QMX_WORKITEMS_SNAPSHOT.WC_BOOL_EXT_TO,
         QMX_WORKITEMS_SNAPSHOT.WC_DOUBLE_EXT_FROM,
         QMX_WORKITEMS_SNAPSHOT.WC_DOUBLE_EXT_TO,
         QMX_WORKITEMS_SNAPSHOT.WC_FLOAT_EXT_FROM,
         QMX_WORKITEMS_SNAPSHOT.WC_FLOAT_EXT_TO,
        QMX_WORKITEMS_SNAPSHOT.WC_INT_EXT_FROM,
        QMX_WORKITEMS_SNAPSHOT.WC_INT_EXT_TO,
        QMX_WORKITEMS_SNAPSHOT.WC_LARGE_STRING_EXT_FROM,
        QMX_WORKITEMS_SNAPSHOT.WC_LARGE_STRING_EXT_TO,
        QMX_WORKITEMS_SNAPSHOT.WC_LONG_EXT_FROM
     TO JAZZ ;

  GRANT DELETE, INSERT, SELECT, UPDATE
     ON TABLE
        QMX_WORKITEMS_SNAPSHOT.WC_LONG_EXT_TO,
        QMX_WORKITEMS_SNAPSHOT.WC_MEDIUM_STRING_EXT_FROM,
        QMX_WORKITEMS_SNAPSHOT.WC_MEDIUM_STRING_EXT_TO,
        QMX_WORKITEMS_SNAPSHOT.WC_STRING_EXT_FROM,
        QMX_WORKITEMS_SNAPSHOT.WC_STRING_EXT_TO,
        QMX_WORKITEMS_SNAPSHOT.WC_TIMESTAMP_EXT_FROM,
        QMX_WORKITEMS_SNAPSHOT.WC_TIMESTAMP_EXT_TO,
        QMX_WORKITEMS_SNAPSHOT.WORKITEM_CHNGS_FROM,
        QMX_WORKITEMS_SNAPSHOT.WORKITEM_CHNGS_TO,
        QMX_WORKITEMS_SNAPSHOT.WS_BIGDECIMAL_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_BOOL_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_DOUBLE_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_FLOAT_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_INT_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_LARGE_STRING_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_LONG_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_MEDIUM_STRING_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_STRING_EXT,
        QMX_WORKITEMS_SNAPSHOT.WS_TIMESTAMP_EXT
     TO JAZZ ;
```

Running UNLOAD and LOAD processes on DB2 to create a backup database

You can use DB2 for z/OS to unload your existing version of each Rational solution for Collaborative Lifecycle Management (CLM) database and use the unloaded data sets to create a backup database.

You can use the unloaded data sets to create a backup database using either an existing database or creating a new database with a new database schema prefix. You can use the backup database you create to restore your database if you have any problems during an upgrade from one version of CLM to another.

To automate the process, the SMP/E package includes programs and JCL samples. Use these programs with the standard DB2 for z/OS utility programs to back up your CLM repositories.

These programs are compatible with Rational Team Concert for System z V2, Rational Team Concert V3, and V3.0.1.x or V4.0 of the Rational solution for Collaborative Lifecycle Management (CLM 3.0.1.x or CLM 4.0); however, the programs are only installed with CLM 3.0.1 and CLM 4.0. You can only unload and load database tables for the same version of the product.

Overview

The process involves generating UNLOAD templates based on the database being unloaded, then running the UNLOAD. The UNLOAD process will unload data in sequential data sets and, in the case of the LOB tables, HFS directories. The process will generate the LOAD utility control statements to be used later in the LOAD process. Part of the process will merge the separate LOAD utility control statements into a single sequential data set and edit this data set to modify some of the load parameters. Then the LOAD runs, and after it runs, the REPAIR utility control statements are generated based on the database being loaded. Finally, a REPAIR step runs.

Important: These tasks should be performed by an experienced DB2 database administrator. Do not attempt to run these jobs if you are not familiar with DB2 utilities. Test this process carefully, in particular the LOAD, with test data or on a test LPAR before using the process on your production data. Back up to a separate DB2 for z/OS database before using any production data.

Shipped components to aid backup and recovery

The following components are included in the SMP/E package in FMID HRJS406.

BLZUNLD
A program in *hlq*.SBLZLOAD that uses parameters that are passed to it to run SQL statements that generate the DB2 for z/OS UNLOAD utility control statements. The UNLOAD utility control statements are required to perform the unload process.

BLZBIND
JCL member in *hlq*.SBLZSAMP that is used to run the DB2 for z/OS BIND for programs BLZUNLD and BLZREPR (sample JCL).

BLZDB2UN
Sample JCL member in *hlq*.SBLZSAMP that is used to unload the CLM DB2 for z/OS database (sample JCL).

BLZSYSIN
Sample REXX member in *hlq*.SBLZSAMP that merges the SYSIN files generated by the UNLOAD process and then completes the following edits on the merged SYSIN file:

1. Change the schema prefix to a new schema prefix if you are loading to a new set of tables.
2. In the LOAD utility control statements, change `RESUME YES` to `RESUME NO REPLACE`.

BLZREPR
A program in *hlq*.SBLZLOAD that uses the DB2 for z/OS database parameters passed to it to run SQL statements that generate the DB2 for z/OS REPAIR utility control statements. The REPAIR utility control statements are required to perform the REPAIRs following the LOAD.

BLZDB2LD
Sample JCL member in *hlq*.SBLZSAMP that is used to load the CLM database tables.

BLZDWHUN
Sample JCL member in *hlq*.SBLZSAMP that is used to unload the CLM data warehouse tables.

BLZDWHLD
Sample JCL member in *hlq*.SBLZSAMP that is used to load the CLM data warehouse tables.

Initial setup required before running the unload process

Before you run the unload process, you must modify and run the BLZBIND job in `hlq.SBLZSAMP` to create the DB2 for z/OS PLAN for the BLZUNLD and BLZREPR programs that will be used in subsequent steps. Follow the instructions included with the job to make the required modifications.

Steps required to unload and reload the DB2 for z/OS tables

Complete the following steps to unload your existing DB2 for z/OS tables and reload a new or existing set of DB2 for z/OS tables. This step does not include the data warehouse tables, which are included in the next section.

Unload the CLM tables (except the data warehouse tables)

Sample jobs are provided that can be used to unload your Change and Configuration Management (CCM), Jazz Team Server (JTS), or Quality Manager (QM) tables. Modify the sample jobs to point to the databases and schema prefixes you require.

The sample jobs provided to perform the UNLOAD task have several steps:

1. Using the parameters provided in the jobs, call program BLZUNLD to generate the unload templates that DB2 for z/OS will use to unload the tables. The tables that contain LOB columns must be handled differently; therefore, these unloads split the UNLOAD utility control statements into two for use in later steps.
2. Print the UNLOAD utility control statements for reference.
3. The LOB tables must be unloaded into a z/OS UNIX Systems Services (USS) directory; therefore, there is a step to create directories for each LOB tablespace. These directories will be used during the UNLOAD and LOAD processes.
4. Unload the non-LOB tables into sequential data sets. These data sets are created using the parameters specified in the job and will have the format of <*hlq*>.<*schema prefix*>.<*tablespace name*>. The high-level qualifier (*hlq*) and schema prefix are passed as parameters to the job.
5. Unload the LOB tables into z/OS USS directories. These directories are created in the MKDIR step based on the **`ussWorkDir`** parameter set in the job.

   Note: It is likely that the backup directories for the LOB tables are large, in which case, create a separate HFS file system mounted to the `ussWorkDir`. HFS file systems have better performance than zFS file systems in this situation.

Complete the following steps to unload your CLM tables:

1. Stop the server.
2. If any of your CLM tables are in DB2 for z/OS, modify and submit the BLZDB2UN job in `hlq.SBLZSAMP` to unload the Jazz Team Server tables. Follow the instructions included with the job to make the required modifications.
3. Run the BLZDB2UN job for each DB2 database you want to unload.

The following two tasks describe how to either load the database tables back to the same database or to a different database.

Load the CLM tables back to the same database

The sample jobs provided to perform the LOAD task have a several steps:

1. Delete the merged SYSIN data set.
2. Using the parameters provided in the job, call program BLZSYSIN to merge the SYSIN data sets that contain the DB2 for z/OS LOAD utility control statements into a single sequential data set. BLZSYSIN also modifies the DB2 for z/OS LOAD parameters of `RESUME YES` to `RESUME NO REPLACE`.
3. Load the DB2 for z/OS tables using the modified SYSIN DB2 for z/OS LOAD utility control statements.
4. Call program BLZREPR to generate DB2 for z/OS REPAIR utility control statements for the next step, based on the parameter passed to the program.
5. Run the DB2 for z/OS REPAIR utility using the utility cards created in previous steps.

Complete the following steps to load your CLM tables back to the same database:

1. Stop the server.
2. Drop the required set of tables. Make sure this process works with loading to a different database before you attempt this procedure. The recommended method for testing is to DROP the database and re-create it with DDL shown in the following example for the Jazz Team Server tables:

   ```
   Drop the following Database for the Jazz Team Server
   DROP DATABASE JTS406;
   COMMIT;

   Create the following Database for the Jazz Team Server
   CREATE DATABASE JTS406 STOGROUP CLMSTG BUFFERPOOL BP16K0
          CCSID UNICODE;
   COMMIT;
   ```

   For additional information about creating the DB2 for z/OS databases, see "Setting up DB2 for z/OS to use with Jazz Team Server" on page 20.
3. Create the new CLM tables by running a **`repotools –createTables`** command for each application (JTS, CCM and QM). To do this, configure and run the BLZCREDB job in *`hlq`*`.SBLZSAMP`, which was run to initially create the tables. This job is used to create the JTS, CCM and QM tables by changing the **`@appl@`** parameter as described in the instructions included with the jobs. For more information on running this job, see "Creating database tables using repository tools" on page 26.
4. Modify and submit the BLZDB2LD job in *`hlq`*`.SBLZSAMP` to load the CLM tables. Follow the instructions included with the job to make the required modifications. Note the parameters passed to the BLZSYSIN module. Because you are using the same schema prefix you will not be required to provide a new schema prefix in variable *@newPrefix@*.
5. Run the BLZDB2LD job for each set of tables you want to load.

Load the CLM tables to a new database

The sample jobs provided to perform the LOAD task have a several steps:

1. Delete the merged SYSIN data set.

2. Using the parameters provided in the job, call program BLZSYSIN to merge the SYSIN data sets that contain the DB2 for z/OS LOAD utility control statements into a single sequential data set. BLZSYSIN also modifies the DB2 for z/OS LOAD parameters of `RESUME YES` to `RESUME NO REPLACE`.
3. Load the DB2 for z/OS tables using the modified SYSIN DB2 for z/OS LOAD utility control statements.
4. Call program BLZREPR to generate DB2 for z/OS REPAIR utility control statements for the next step based on the parameter passed to the program.
5. Run the DB2 for z/OS REPAIR utility using the utility cards created previously.

Complete the following steps to load your CLM tables:

1. Stop the server.
2. Create the JTS and CCM tables as documented in "Setting up DB2 for z/OS to use with Jazz Team Server" on page 20. Note that you are creating a new set of tables with a new schema prefix, so you must use a modified `teamserver.properties` file that uses the correct schema prefix. Save the old `teamserver.properties` file so you can use it with the existing set of tables. Ensure that you use the new, modified `teamserver.properties` file or the **`–createTables`** command will delete the existing data.
3. Create the new CLM tables by running a **`repotools –createTables`** command for each application (JTS, CCM and QM). To do this, configure and run the BLZCREDB job in *`hlq`*`.SBLZSAMP`, which was run initially to create the tables. This job is used to create the JTS, CCM and QM tables by changing the **`@appl@`** parameter as described in the instructions included with the jobs. For more information on running this job, see "Creating database tables using repository tools" on page 26.
4. Modify and submit the BLZDB2LD job in *`hlq`*`.SBLZSAMP` to load the CLM tables. Follow the instructions included with the job to make the required modifications. Note the parameters passed to the BLZSYSIN module. Because you are not using the same schema prefix you must provide a new schema prefix in variable *@newPrefix@*. The first step in this job will modify the LOAD utility control statements for you to set the schema prefix to the new value. The**`@dbname@`** parameter represents the name of the new database into which you will LOAD the tables.
5. Run the BLZDB2LD job for each set of tables you want to load.

Steps required to unload and reload the DB2 for z/OS data warehouse tables

If you have the data warehouse tables installed there are sample jobs provided that you can use to unload and load the data warehouse tables. Modify the sample jobs to point to the databases and schema prefixes you require.

Unload the CLM data warehouse tables

The sample jobs provided to perform the UNLOAD task have a single step:

1. Unload the data warehouse tables into sequential data sets. These data sets are created using the parameters specified in the job and will have

the format of: `<hlq>.DWH.<tablespace_name>`. The high-level qualifier (hlq) is passed as parameters to the job.

Note: Do not change the middle-level qualifier, DWH, because it is used by both the UNLOAD and LOAD jobs.

Complete the following steps to unload your CLM tables:

1. Stop the server.
2. If any of your data warehouse tables are in DB2 for z/OS, modify and submit the BLZDWHUN job in *hlq*.SBLZSAMP to unload the Jazz data warehouse tables. Follow the instructions included with the job to make the required modifications.

Load the CLM tables back to the same database

The sample jobs provided to perform the LOAD task have a several steps:

1. Delete the merged SYSIN data set.
2. Using the parameters provided in the job, call program BLZSYSIN to merge the SYSIN data sets that contain the DB2 for z/OS LOAD utility control statements into a single sequential data set. BLZSYSIN also modifies the DB2 for z/OS LOAD parameters of `RESUME YES` to `RESUME NO REPLACE ENFORCE NO`.
3. Load the DB2 for z/OS tables using the modified SYSIN DB2 for z/OS LOAD utility control statements.
4. Call program BLZREPR to generate DB2 for z/OS REPAIR utility control statements and DB2 for z/OS CHECK utility control statements for the next steps based on the parameter passed to the program.
5. Run the DB2 for z/OS CHECK utility using the utility cards created in the previous step to check referential integrity.
6. Run the DB2 for z/OS REPAIR utility using the utility cards created in previous steps.

Complete the following steps to load your data warehouse tables:

1. Stop the server.
2. Drop the required set of tables. The recommended method is to DROP the database and re-create it with DDL shown in the following example for the data warehouse tables:

   ```
   Drop the following Database for the Jazz Team Server
   DROP DATABASE DWH40;
   COMMIT;

   Create the following Database for the Jazz Team Server
   CREATE DATABASE DWH40 STOGROUP CLMSTG BUFFERPOOL BP16K0
           CCSID UNICODE;
   COMMIT;
   ```

   For additional information about creating the DB2 for z/OS databases, see “Setting up DB2 for z/OS to use with Jazz Team Server” on page 20.
3. Create the new CLM tables by running a **`repotools –createWarehouse`** command for the JTS application. To do this, configure and run the BLZCREDB job in *hlq*.SBLZSAMP, which was run to initially create the tables. This job is used to create the JTS tables by changing the **`@appl@`** parameter as described in the instructions included with the jobs. For more information on running this job, see “Creating database tables using repository tools” on page 26.

4. Modify and submit the BLZDWHLD job in *hlq*.SBLZSAMP to load the data warehouse tables. Follow the instructions included with the job to make the required modifications. Note the parameters passed to the BLZSYSIN module.

Load the CLM tables to a new database

The sample jobs provided to perform the LOAD task have a several steps:

1. Delete the merged SYSIN data set.
2. Using the parameters provided in the job, call program BLZSYSIN to merge the SYSIN data sets that contain the DB2 for z/OS LOAD utility control statements into a single sequential data set. BLZSYSIN also modifies the DB2 for z/OS LOAD parameters of `RESUME YES` to `RESUME NO REPLACE ENFORCE NO`.
3. Load the DB2 for z/OS tables using the modified SYSIN DB2 for z/OS LOAD utility control statements.
4. Call program BLZREPR to generate DB2 for z/OS REPAIR utility control statements and DB2 for z/OS CHECK utility control statements for the next steps, based on the parameter passed to the program.
5. Run the DB2 for z/OS CHECK utility using the utility cards created in previous step to check referential integrity.
6. Run the DB2 for z/OS REPAIR utility using the utility cards created previously.

Complete the following steps to load your data warehouse tables

1. Stop the server.
2. Create the data warehouse tables according to the instructions in "Setting up DB2 for z/OS to use with Jazz Team Server" on page 20.

   Note: You are creating a new set of tables, so you must use a modified `teamserver.properties` file that uses the data warehouse database.
3. Create the new CLM tables by running a **`repotools –createWarehouse`** command for the JTS application. To do this, configure and run the BLZCREDB job in *hlq*.SBLZSAMP, which was run to initially create the tables. This job is used to create the JTS tables by changing the **`@appl@`** parameter as described in the instructions included with the jobs. For more information on running this job, see "Creating database tables using repository tools" on page 26.
4. Modify and submit the BLZDWHLD job in *hlq*.SBLZSAMP to load the data warehouse tables. Follow the instructions included with the job to make the required modifications. Note the parameters passed to the BLZSYSIN module.

Deploying and starting the server on z/OS

Installations on z/OS systems have several database and application server options.

Before you begin

Follow the instructions in the following sections according to the database and application server options you are using:

- "Running Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications on z/OS with Tomcat and Derby" on page 36

- “Running Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications on z/OS with Tomcat and DB2” on page 37
- “Running Jazz Team Server and the applications for Rational solution for Collaborative Lifecycle Management applications on a z/OS system with WebSphere Application Server” on page 38

Running Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications on z/OS with Tomcat and Derby

Running the Jazz Team Server and the Rational solution for Collaborative Lifecycle Management (CLM) applications on z/OS with Apache Tomcat and Derby involves several sample JCL members and configuration property files.

Before you begin

In order to run Tomcat, first you must have SMP/E applied FMID HROC406 (optional components) and run configuration job `BLZCPOPT`. The `BLZCPOPT` configuration job populates and configures the Tomcat directories. You must also run BLZCP* jobs for JTS, CCM, QM, and RM depending on which applications you plan to deploy. For additional information on running the BLZCPOPT job and other BLZCP* jobs, see “Creating directories for Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications” on page 15.

Reviewing the Apache Tomcat port configuration

To avoid port conflicts, review the Tomcat application server default port settings to make sure that they are not the same as other application server instances that you have installed.

About this task

By default, the Tomcat application server uses a nonsecure port 9080 and secure port 9443, along with ports 9005, 9009, and 8082. These default ports might conflict with a default WebSphere Application Server instance that you have installed. Review the Tomcat server.xml file, which defines these ports and is located at `@workPath@/catalina_base/conf/server.xml`.

Tip: If you modify the default ports in the server.xml file, record the changes so you can use these ports in other configuration tasks.

Choose one of the following options based on the application server and database you are using:

- Running the Jazz Team Server for System z with Tomcat and Derby
- Running the Jazz Team Server for System z with Tomcat and DB2 for z/OS

Starting Tomcat using the Derby database

Before you start Tomcat, you must customize the environment for any Rational solution for Collaborative Lifecycle Management (CLM) applications you plan to run.

Before you start Tomcat, you must complete BLZCPOPT and the BLZCP* customization jobs for JTS or any CLM applications you plan to run. See “Creating directories for Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications” on page 15 for details.

After you run BLZCP* jobs, configure member `BLZSRVC` in *hlq*.`SBLZSAMP` using the instructions contained in the sample. Use the **SUBMIT** command to submit the modified JCL and check the job log carefully for errors and messages that the server startup has completed. You can adjust the WLM Service Class for the `BLZSRVC` job to meet your system requirements.

You can stop the Jazz Team Server by stopping this job.

Running Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications on z/OS with Tomcat and DB2

Running Jazz Team Server and the Rational solution for Collaborative Lifecycle Management (CLM) applications on z/OS with Apache Tomcat and DB2 for z/OS involves several sample JCL members and configuration property files.

Before you begin

Before you can run Tomcat, you must complete the following tasks:

- Install FMID HROC406 (optional components) from the SMP/E package.
- Run configuration jobs BLZCPOPT, BLZCPJTS, and BLZCP* for any Rational solution for Collaborative Lifecycle Management (CLM) application you are installing. See "Creating directories for Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications" on page 15 for details.

  Note: The BLZCPOPT configuration job populates and configures the Tomcat directories.
- Create DB2 for z/OS repositories and database tables to support your configuration. See "Setting up a DB2 database on a z/OS system" on page 19 for details.

Reviewing the Apache Tomcat port configuration

To avoid port conflicts, review the Tomcat application server default port settings to make sure that they are not the same as other application server instances that you have installed.

About this task

By default, the Tomcat application server uses a nonsecure port 9080 and secure port 9443, along with ports 9005, 9009, and 8082. These default ports might conflict with a default WebSphere Application Server instance that you have installed. Review the Tomcat server.xml file, which defines these ports and is located at `@workPath@/catalina_base/conf/server.xml`.

Tip: If you modify the default ports in the server.xml file, record the changes so you can use these ports in other configuration tasks.

Choose one of the following options based on the application server and database you are using:

- Running the Jazz Team Server for System z with Tomcat and Derby
- Running the Jazz Team Server for System z with Tomcat and DB2 for z/OS

Starting Apache Tomcat server by using the DB2 for z/OS database

You can start Tomcat by using the DB2 for z/OS database.

You are ready to start Tomcat by using the DB2 for z/OS database. If you must run other Jazz Team Server repository tools functions, like importing or exporting data for migration or re-creating a new database, see “Creating database tables using repository tools” on page 26.

Configure member `BLZSRVD` in *`hlq`*`.SBLZSAMP` by using the instructions that are contained in the sample. Use the **`SUBMIT`** command to submit the modified JCL and check the job log carefully for errors and messages that the server startup was completed. You can adjust the WLM Service Class for the `BLZSRVD` job to meet your system requirements.

You can stop the Jazz Team Server by stopping this job.

Running Jazz Team Server and the applications for Rational solution for Collaborative Lifecycle Management applications on a z/OS system with WebSphere Application Server

Running the Jazz Team Server and the Rational solution for Collaborative Lifecycle Management (CLM) applications on a z/OS system with WebSphere Application Server involves setting up security and configuring the WebSphere Application Server.

About this task

These instructions supplement the basic considerations for deploying and starting WebSphere Application Server.

You can run the CLM applications with an existing instance of WebSphere Application Server for z/OS if it meets the system requirements

- WebSphere Application Server OEM for z/OS Program Directory at http://www.ibm.com/support/docview.wss?uid=swg27021864
- WebSphere Application Server OEM Edition for z/OS Configuration Guide at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=GA320631

Setting up user security on a z/OS system by using RACF

Additional information you need for setting up user security on a z/OS system.

Before you begin

Review the information in Permissions at http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0m6/topic/com.ibm.jazz.platform.doc/topics/c_permissions.html in the Rational solution for Collaborative Lifecycle Management information center.

About this task

Repository permissions and role-based permissions for users are similar for servers running on z/OS and other platforms; however, on z/OS, when the Jazz Team Server and the Rational solution for Collaborative Lifecycle Management (CLM) applications are configured to use RACF for security, the WebSphere Application Server determines repository permissions based on RACF EJBROLE profiles for security control.

Define the Jazz Team Server and CLM application repository permissions using the `EJBROLE` class by completing the following tasks:

1. Define the `EJBROLE` profiles:

   JazzAdmins
   Jazz repository administrators with full read/write access.

   JazzDWAdmins
   Jazz repository administrators with specific permissions to control the data warehouse on a Jazz Team Server.

   JazzProjectAdmins
   Jazz repository administrators with specific permissions to manipulate project areas, team areas, and process templates.

   JazzGuests
   Users with read-only access to the Jazz repository.

   JazzUsers
   Users with regular read/write access to the Jazz repository.

   Example RACF commands:

   ```
   RDEFINE EJBROLE JazzAdmins UACC(NONE)
   RDEFINE EJBROLE JazzDWAdmins UACC(NONE)
   RDEFINE EJBROLE JazzProjectAdmins UACC (NONE)
   RDEFINE EJBROLE JazzGuests UACC(READ)
   RDEFINE EJBROLE JazzUsers UACC(NONE)
   ```

2. Permit the appropriate access to users or groups.

   Example RACF commands:

   ```
   Permit JazzAdmins CLASS(EJBROLE) ID(jazAdmns) ACCESS(READ)
   Permit JazzDWAdmins CLASS(EJBROLE) ID(jDwadmns) ACCESS(READ)
   Permit JazzProjectAdmins CLASS(EJBROLE) ID(jPradmns) ACCESS (READ)
   Permit JazzUsers CLASS(EJBROLE) ID(jazzgrp) ACCESS(READ)
   ```

3. Activate the new definitions:

   After the RACF RDEFINE and PERMIT commands, you must issue the following command to take them into account:

   ```
   SETROPTS RACLIST(EJBROLE) REFRESH
   ```

4. After you configure Jazz Team Server, you must log on as a Jazz Team Server administrator to verify the configuration. Before attempting to verify the configuration, provide at least one user ID or group with read authority to the JazzAdmins profile in the EJBROLE class.

   Notes:

   - When you add user IDs to the Jazz repository, you must also give them read authority to the appropriate RACF profile in the EJBROLE class (JazzAdmins, JazzDWAdmins, JazzProjectAdmins, JazzGuests, JazzUsers).
   - Specifying a System Authorization Facility (SAF) profile prefix during the WebSphere Application Server customization process affects the way the EJBROLE profiles are referenced. For more information on using EJBROLE profiles, see the WebSphere Application Server Version 7.0 Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp and search for *System Authorization Facility for role-based authorization*.
   - If you select the **Enable SAF Authorization** option as the external authorization provider and select **Use the APPL profile to restrict**

access to the application server, you must grant READ access to the APPL profile for the Rational Team Concert users. See the WebSphere Application Server Version 7.0 Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp for more information.

- If it is set, note the value of `com.ibm.security.SAF.profilePrefix`. It will be used as a prefix for the EJBROLEs in RACF.

Attention: When your password expires, you cannot connect to the Jazz Team Server, but you will **not** receive an error message that indicates your password has expired. If you cannot connect to the Jazz Team Server because of an expired password, you must change the password using TSO or Rational Developer for System z.

Configuring Jazz Team Server and the applications for the Rational solution for Collaborative Lifecycle Management to use WebSphere Application Server on z/OS

This topic describes how to set up the Jazz Team Server and the Rational solution for Collaborative Lifecycle Management (CLM) applications to work with the WebSphere Application Server on z/OS. Configuring the WebSphere Application Server on z/OS is similar to other platforms. The primary differences are the security model and the locations of files in the file system.

Before you begin

Before you perform this task, make sure you:

- Installed and are running the WebSphere Application Server.
- Customized and submitted the BLZCP* jobs for the Jazz Team Server and any CLM applications you plan to deploy.
- Created the DB2 databases and updated the `teamserver.properties` files with your database settings.
- Selected and configured **Local Operating System**, if you are using RACF security.
- Established RACF security settings. For more information on RACF security, see "Setting up user security on a z/OS system by using RACF" on page 38.
- Turned off the Java 2 security option by following these steps:
  1. Click **Security** > **Global security**.
  2. Under Java 2 security, clear the check box for **Use Java 2 security to restrict application access to local resources**.
  3. Ensure that the check boxes for **Enable administrative security** and **Enable application security** are selected.

  If this option is turned on in the WebSphere Application Server, the web application will not start.
- Selected **Use available authentication data when an unprotected URI is accessed**. If you are using the Integrated Solutions Console for the server, use the following steps to verify this setting:
  1. Click **Security** > **Global security** > **Web and SIP security** > **General settings**.
  2. Select the check box for **Use available authentication data when an unprotected URI is accessed**.
- Verified or updated the WebSphere Application Server level. Jazz Team Server requires that WebSphere Application Server for z/OS v8.0.0.3 or WebSphere Application Server OEM Edition for z/OS v7.0 Fixpack 15.

About this task

This topic does not list the steps for configuring the WebSphere Application Server authentication on SSL. For detailed information about the various authentication and encryption options for WebSphere Application Server, search for *Securing applications and their environment* in the appropriate the information center for the version you are running:

- Version 8.0: http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp
- Version 7.0: http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp

Procedure

Perform the following steps to deploy the Jazz Team Server and CLM applications on the WebSphere Application Server. Each step is detailed below in its own task.

1. Define application server JVM custom properties.
2. Deploy the web applications and restart the application server.

Defining application server JVM custom properties:
About this task

Notes:

- There might be minor differences in the WebSphere administrative console depending on your version of the WebSphere Application Server.
- Substitute *@confPath@* with your Jazz Team Server configuration directory as determined when you ran the BLZCPJTS and other BLZCP* sample jobs for the CLM applications.

Procedure

1. Log in to the WebSphere Application Server administrative console.
2. Select **Servers** > **Application Servers**.
3. Select **server1** or your server name.
4. Expand **Container Services**.
5. Select **ORB Service**.
6. Select **z/OS Additional Settings**.
7. Set **Workload Profile** to *LONGWAIT*.
8. Click **OK** and save the changes.
9. Select **Servers** > **Application Servers**.
10. Select **server1** or your server name.
11. Select **Java and Process Management** > **Process Definition** > **Servant**.
12. Select **Java Virtual Machine**.
13. Set the value for the **Initial Heap Size** to 4096.
14. Set the value for **Maximum Heap Size** to 4096. This is a suggested value. Set the value according to your environment. A typical medium-sized team can use a value of 4096, providing 4 GB of heap memory for the Jazz Team Server process using a 64-bit JVM.
15. Set the value for **Generic JVM arguments** to `-Xmn512m`.
16. Click **Apply**.
17. Select **Custom Properties**.
18. Add the following properties by clicking **New**. Click **OK** after you add each one.

Name: **`java.awt.headless`**
Value: `true`

Name: **`org.eclipse.emf.ecore.plugin.EcorePlugin.doNotLoadResourcesPlugin`**
Value: `true`

Name: **`JAZZ_HOME`**
Value:`file:///etc/jazz406` or whatever value you have used for *@confPath@* during initial set up.

Name: **`DB2Z_JDBC`**
Value: `/`*`@db2PathPrefix@`*`/db2910_jdbc/classes` or `/`*`@db2PathPrefix@`*`/db2a10/jdbc/classes`

Note: Replace *@db2PathPrefix@* with the path to the DB2 V9 or V10 JDBC license and JDBC jar files on your system. These files are named `db2jcc_license_cisuz.jar` and `db2jcc.jar` and are located by default at `/usr/lpp/` plus the version-specific path.

Name: **`log4j.configuration`**
Value: `file:///etc/jazz406/startup_log4j.properties`

Note: The settings contained in `startup_log4j.properties` are used at the early stages in the startup process to print messages to the WebSphere Application Server `SystemOut.log` file. After the early stages, each Jazz application will switch to using the application-specific settings from `/etc/jazz406/`*`<app context>`*. If you use a configuration path other than `/etc/jazz406`, adjust this path accordingly.

19. Select **Save directly to the master configuration**.

Deploying the web applications and restarting the WebSphere Application Server:
Procedure

1. From the WebSphere Application Server administrative console, select **Applications** > **Install New Application** (or **Applications** > **New Application** > **New Enterprise Application**).
2. Under *Path to new Application*, select **Remote file system**.
3. Under *Full path*, browse to *`@pathPrefix@`*`/usr/lpp/jazz/v4.0.6/server/tomcat/webapps/jts.war`, where *@pathPrefix@* is any prefix you specified when you installed the SMP/E.
4. Set the context root to `/jts`.
5. Click **Next**, then continue with the remaining *Install New Application* steps as prompted by that editor.
6. Click **Finish**.
7. Select **Save directly to the master configuration**.
8. Repeat the previous steps for the `admin.war` with context root `/admin`.
9. Repeat the previous steps for the `clmhelp.war` with context `root/clmhelp`.
10. If you are installing the CCM application, repeat the previous steps for the `ccm.war` with context root `/ccm`.
11. If you are installing the QM application, repeat the previous steps for the `qm.war` with context root `/qm`.
12. If you are installing the RM application, repeat the previous steps for the `rm.war` with context root `/rm`.

13. Restart the application server.
14. Check the servant logs and system console for errors.

What to do next

After you start the server, review this topic: Running the setup wizard in the Rational solution for Collaborative Lifecycle Management information center.

Chapter 3. Installing optional tools, clients, and samples on z/OS

Installing and configuring the Build System Toolkit on z/OS systems

The Build System Toolkit, which is an optional component, can be installed on z/OS by using the SMP/E installation process.

By default, the SMP/E installation process places the Build System Toolkit in the following directory: `@pathPrefix@/usr/lpp/jazz/v4.0.6/buildsystem` and in the following libraries: SBLZLOAD (load modules) and SBLZMxxx (ISPF messages).

For detailed installation instructions, see "SMP/E installation process" on page 1.

Creating additional Build System Toolkit directories

On z/OS, the Build System Toolkit requires several directories in addition to the z/OS UNIX System Services directory created by the SMP/E installation. One directory is a working directory and the other directories are used to store configuration files. There is a sample member (`BLZCPBTK`) in *`hlq`*`.SBLZSAMP` used to create and populate these directories with the required files from the SMP/E installed directory.

The following three symbolic names are used throughout this guide to indicate the following directories:

Table 4. Symbolic names for directories

Symbolic name	Use	Variable in BLZCP* jobs	Default directory
@pathPrefix@	The path prefix specified during the SMP/E installation.	BLZHOME	
@confPath@	The Jazz Team Server configuration files directory.	BLZCONF	`/etc/jazz406`
@workPath@	The Jazz Team Server working directory.	BLZWORK	`/u/jazz406`

Note: The configuration and work directories require RACF file access permissions. For details, see "Security on z/OS systems" on page 5.

In addition to creating and populating these directories, the sample jobs provided also perform several required customizations. The variables mentioned in the symbolic name table, must be set in several places in each job. In addition there are other variables that might need to be set that are listed in the following table:

Table 5. Variable names and usages

Variable Name	Use	Default value
BLZBFAH	The path prefix of the Buildforge agent directory /buildagent	`/usr/lpp/jazz/v4.0.6/buildagent`

Table 5. Variable names and usages (continued)

Variable Name	Use	Default value
BLZBFAC	The Build Forge agent configuration files directory	`/etc/jazz406/ccm`
BLZJAVA	The location of the Java directory	`/usr/lpp/java/J6.0_64`
iconvLoc	The location of the iconv utility	`/bin/iconv`

The instructions contained in each configuration job will indicate which variables you must configure.

Configure member `BLZCPBTK` in *`hlq`*`.SBLZSAMP` using the instructions contained in the member. Use the SUBMIT command to submit the modified JCL and check the job log. Return codes of 0 indicate the configuration is correct.

Attention: The `BLZCP*` members contain the *@confgrp@* and *@workgrp@* variables, which must be set to the SAF groups that contain all of the user IDs that require write access to the configuration and work directories. You must also associate group IDs (GID) with these SAF groups.

Configuring the ISPF gateway for build, deployment, and promotion support

With Rational Team Concert, you can run programs and executable files as part of build, deployment, and promotion. If these programs or executable files are ISPF-based, they must be started through the ISPF gateway. How these programs and executable files are run depends on the environment from which they are started. You can configure the ISPF gateway to support build, deployment, and promotion components that are available with a Developer for IBM Enterprise Platforms client access license.

Note: For the ISPF gateway component to function correctly, there are APARs that you must install. For more information, see the System requirements for the CLM 2013 releases.

The ISPF gateway enables client applications to connect to a z/OS host and run TSO and ISPF commands. The ISPF gateway is installed as part of ISPF as load modules and files in the HFS. These files are typically installed in `/usr/lpp/ispf/bin`. The build, deployment, and promotion components are created as part of the SMP/E installation. Depending on your setup, extra modifications might be required of the ISPF gateway supplied scripts and components. The default settings and procedure for modifying the defaults are described later.

The following table shows how programs or executable files can be run during build, deployment, and promotion, and the available support for them.

Table 6. Starting commands for Rational Team Concert components

Definition	Command	Started from
Build translator	ISPF command or executable file	ISPF gateway
	TSO command or executable file	ISPF gateway

Table 6. Starting commands for Rational Team Concert components (continued)

Definition	Command	Started from
Build definition	Pre-build command line	UNIX System Services (USS) command
	Post-build command line	UNIX System Services (USS) command
Package definition	Package pre-command	ISPF gateway
	Package post-command	ISPF gateway
Deployment definition	Deploy pre-command	ISPF gateway
	Deploy post-command	ISPF gateway
	Rollback pre-command	ISPF gateway
	Rollback post-command	ISPF gateway
Promotion definition	Pre-build command line	UNIX System Services (USS) command
	Post-build command line	UNIX System Services (USS) command

Programs or executable files can be run by using the ISPF gateway or a UNIX System Services (USS) command. USS commands cannot start ISPF commands, and they can only start limited TSO commands. Commands started through the ISPF gateway have the full set of ISPF and TSO APIs available. Pre-build and post-build commands for build and promotion definitions are started through USS. For more information, see "Starting the ISPF gateway from a USS command" on page 51.

The following components are included with the ISPF gateway:

ISPF gateway environment file: `ISPZXENV`

ISPF gateway environment file that contains customizable settings for the ISPF gateway.

The `ISPZXENV` file is provided as part of z/OS. By default this file is installed into `/usr/lpp/ispf/bin`. However, this directory is marked READ only after installation. This environment file is run by the ISPF gateway to get the value of certain environment variables necessary for the ISPF gateway to run. Default variables include the following:

STEPLIB

By default is set to `STEPLIB = 'ISP.SISPLPA:ISP.SISPLOAD'`

Note: You can add more system load libraries to this STEPLIB allocation. For example, DB2.SDSNLOAD, if you have programs or executable files that start DB2 functions such as BIND.

CGI_ISPCONF

By default is set to `CGI_ISPCONF = '/etc/ispf'`

CGI_ISPWORK

By default is set to `CGI_ISPWORK = '/var/ispf'`

To set variables to non-default values, copy `ISPZXENV` to a writable configuration directory, such as `/etc/ispf` and set the variables to your required values. You must then alter your path environment variable in `Startispf.sh`, which is described later, to read that location before the default ISPF bin path.

For example:

```
export PATH=/etc/ispf:$PATH
```

In the ISPF gateway work directory, by default `/var/ispf`, you must have an existing directory created called `WORKAREA`, which has read and write permissions for all users. Temporary directories of the format `/var/ispf/WORKAREA/<userid>/*` are created when the ISPF gateway is used. If you are using the default ISPF gateway installation as provided by ISPF, this directory is typically in `/var/ispf/WORKAREA`. If the directory does not exist, follow the instructions in the chapter that is titled, *Installing and customizing the gateway* in ISPF Planning and Customizing (GC34-4814) at http://publibfp.dhe.ibm.com/epubs/pdf/ispzpc80.pdf.

ISPF gateway startup script: `startispf.sh`

Shell script to start the ISPF gateway for deployment and promotion.

In order to start the ISPF gateway, Rational Team Concert provides a shell script that is called from the deployment and promotion Ant tasks. This script is installed into `/usr/lpp/jazz/v4.0.6/buildsystem/buildtoolkit/examples/ispfgateway`. As part of standard customization, this script is copied to the Rational Team Concert configuration directory, typically `/etc/jazz406/ccm` by the `BLZCPBTK` job. Depending on your setup, you might need to modify this shell script to set some environment variables.

PATH The *PATH* variable must point to the ISPF gateway home directory:

```
export PATH=$PATH:/usr/lpp/ispf/bin
```

If you modified `ISPZXENV` you might need to set the PATH to:

```
export PATH=$PATH:/etc/ispf:/usr/lpp/ispf/bin
```

If you use different library concatenations in `ISPF.conf`, configure your installation to support multiple projects and streams by creating multiple copies of the `startispf.sh` script, `ISPF.conf` file, and `ISPZXENV` file. In each copy of the `ISPZXENV` file, set the *CGI_ISPFCONF* variable to point to the location of the related `ISPF.conf` file. Then, in each copy of the `startispf.sh` script, set the *PATH* variable to point to the appropriate copy of the `ISPF.conf` file. Then, use the `startispf.sh` script that corresponds to the environment needed for a particular project.

Enterprise Extensions build startup script: `startbfa.sh`

Shell script to start the Rational Build Agent by using an Enterprise Extensions build.

Enterprise Extensions builds provide a mechanism to start a build that in turn starts a program. The program can be an ISPF command or executable file or a TSO command or executable file. If an ISPF allocated library object is being started by the build , you must configure the build to provide access to the required object. In `startbfa.sh`, you must list the location of the ISPF-supplied files so the Rational Build Agent can start the ISPF gateway. By default these files are in `/usr/lpp/ispf` and are specified in `startbfa.sh` using variable *_CMDSERV_BASE_HOME*. If you have the ISPF gateway binary files installed in another location, you must change this variable. For example:

```
#
# Specify the home directory for the ISPF programs.
#
export _CMDSERV_BASE_HOME=/usr/lpp/ispf
```

ISPF programs and executable files must be allocated in the ISPF gateway configuration file (`ISPF.conf`) as described later. The Rational Build Agent also requires access to the `ISPF.conf`. Access is set in the ISPF gateway environment file (`ISPZXENV`). If you plan to use the ISPF capabilities in the builds and you do not use the default location `/usr/lpp/ispf/bin`, you must create a copy of `ISPZXENV` in a writeable configuration directory and point the Rational Build Agent to that location. Set the build property **`team.enterprise.build.ant.myISPFBinPath`** to the directory location that contains the `ISPZXENV` file. You can edit the value or add the property by using the **Properties** tab in your build definition. For more information, see Managing z/OS builds at http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0m6/topic/com.ibm.team.build.doc/topics/c_rtcz_buildingoverview.html in the Rational solution for Collaborative Lifecycle Management information center.

ISPF gateway configuration file: `ISPF.conf`

ISPF gateway configuration file that contains customizable settings for the ISPF gateway.

The `ISPF.conf` file is provided as part of z/OS. The ISPF gateway configuration file provides the allocations that are required to start an ISPF session in a created task. The `ISPF.conf` file performs a similar function to a logon procedure that allocates the ISPF environment for a logged on ISPF user. In order for the build, deployment, and promotion functions to start modules to perform specific tasks, the modules must be made available in the ISPF concatenations. By default these allocations are set up as follows:

```
sysproc=ISP.SISPCLIB
sysexec=ISP.SISPEXEC
ispmlib=ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
ispllib=ISP.SISPLOAD
```

For deployment and promotion to work, add the Rational Team Concert product load library (*hlq*.SBLZLOAD) and product message library (*hlq*.SBLZMENU) to the ISPLLIB and ISPMLIB concatenation. The modified allocations look similar to this example:

```
sysproc=ISP.SISPCLIB
sysexec=ISP.SISPEXEC
ispmlib=BLZ.SBLZMENU,ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
ispllib=BLZ.SBLZLOAD,ISP.SISPLOAD
```

During build, promotion, or deployment, if you run your own programs or REXX executable files and those programs or executable files are called as an ISPF command, for example **`TSO DOSTUFF`**, that is not fully qualified, such as `EX 'MYPROJ.MYREXX(DOSTUFF)'`, add the programs or executable files to the ISPF concatenation in the `ISPF.conf` file. In the `ISPF.conf` file, make load module data sets available in the ISPLLIB concatenation, and make executable files available in the SYSPROC or SYSEXEC concatenations. For example:

```
sysproc=ISP.SISPCLIB,MYPROJ.MYCLIST
sysexec=ISP.SISPEXEC,MYPROJ.MYREXX
ispmlib=BLZ.SBLZMENU,ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
ispllib=BLZ.SBLZLOAD,ISP.SISPLOAD,MYPROJ.LOAD
```

Running interpretive deployment modules

If you want to add your own deployment processes or functions, you can run interpretive versions of the deployment modules. The deployment component runs the following programs:

Table 7. Programs run by the deployment component

Program	Function	Description
BLZPKGZP	Deployment packaging	Using the manifest file create a .zip file of the specified contents.
BLZBKPZP	Deployment backup	Using the package file, make backup copies of the modules the deployment overwrites.
BLZDEPZP	Deployment	Using the deployment manifest file, deploy the contents of the specified .zip file.

The Rational Team Concert deployment tasks call these programs to perform the described function. If you want to modify these programs you can do so by modifying the supplied samples that are in library *hlq*.SBLZSAMP and then point to the modified version in the ISPF.conf. So if you make the modifications directly in *hlq*.SBLZSAMP your ISPF.conf allocations are similar to this example:

```
sysproc=ISP.SISPCLIB,BLZ.SBLZSAMP
sysexec=ISP.SISPEXEC
ispmlib=BLZ.SBLZMENU,ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
ispllib=BLZ.SBLZLOAD,ISP.SISPLOAD
```

ISPF checks the ISPLLIB concatenation first unless the command has percent symbol % prefix; therefore, you must rename or remove the modules you changed from the *hlq*.SBLZLOAD data set.

Adding your own EXEC libraries to ISPF gateway configuration file

For Enterprise Extensions builds, if you use the ISPF call method, you must also add your own library of executable files to the SYSEXEC or SYSPROC concatenations in the ISPF.conf. If you specified an executable file name or specified a **SELECT CMD(executable file)**, then ISPF requires access to the required executable file in the appropriate concatenation. You must modify the ISPF.conf to include any required executable file libraries, for example:

```
sysproc=ISP.SISPCLIB
sysexec=ISP.SISPEXEC,MY.PROJECT.EXEC
ispmlib=BLZ.SBLZMENU,ISP.SISPMENU
isptlib=ISP.SISPTENU
ispplib=ISP.SISPPENU
ispslib=ISP.SISPSLIB
ispllib=BLZ.SBLZLOAD,ISP.SISPLOAD
```

For more information about the ISPF gateway, see the chapter titled, *TSO/ISPF client gateway in ISPF Planning and Customizing* in ISPF Planning and Customizing (GC34-4814) at http://publibfp.dhe.ibm.com/epubs/pdf/ispzpc80.pdf.

Starting the ISPF gateway from a USS command

If you need to add a build or promotion definition pre-build or post-build command, and that command requires TSO or ISPF services, start the ISPF gateway from the USS command. The `BLZGTWY` member in *hlq*.`SBLZSAMP` is a sample executable file for starting the ISPF gateway. You can run this executable file from USS command invocation points or from an XML file containing Ant macros. Using the instructions contained in the `BLZGTWY` member, you might need to customize the `/tmp` directory specification and the *PATH* variable to point to where you installed the ISPF gateway code.

The sample executable file `BLZGTWY` accepts the following parameters:

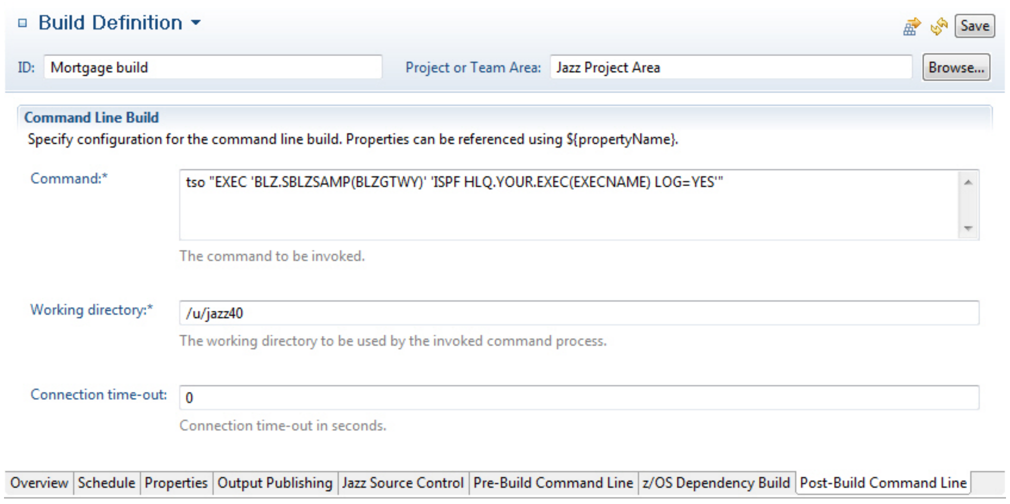
Method
: Invocation method, either TSO or ISPF. The TSO method only allocates SYSPROC and SYSEXEC from the `ISPF.conf` file. The ISPF method allocates all the ISPF concatenation from the `ISPF.conf` file.

Exec
: Name of the executable file to run.

Log
: Optional. Specifying `LOG=YES` writes the full ISPF gateway log, which is useful for debugging problems.

Starting the ISPF gateway from pre- or post-build and pre- or post-promotion commands

To run your own executable file from a pre- or post-build and pre- or post-promotion command, you must call the `BLZGTWY` executable file and pass your own executable file as a parameter. `BLZGTWY` starts the ISPF gateway by passing your executable file to be run in a full ISPF environment. In the following examples, `BLZGTWY` is located in the supplied `SBLZSAMP` library. The following example shows how to start the `BLZGTWY` executable file in a post-build command:



ID: Mortgage build
Project or Team Area: Jazz Project Area

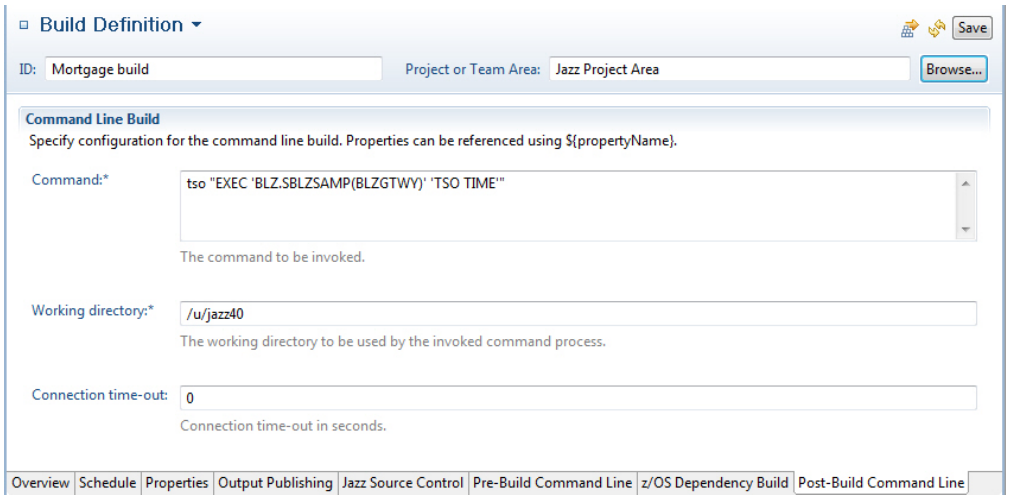
Command Line Build
Specify configuration for the command line build. Properties can be referenced using ${propertyName}.

Command:* tso "EXEC 'BLZ.SBLZSAMP(BLZGTWY)' 'ISPF HLQ.YOUR.EXEC(EXECNAME) LOG=YES'"
The command to be invoked.

Working directory:* /u/jazz40
The working directory to be used by the invoked command process.

Connection time-out: 0
Connection time-out in seconds.

The following example shows how to start the TSO time function with a USS command:



ID: Mortgage build

Project or Team Area: Jazz Project Area

Command Line Build

Specify configuration for the command line build. Properties can be referenced using ${propertyName}.

Command:* tso "EXEC 'BLZ.SBLZSAMP(BLZGTWY)' 'TSO TIME'"

The command to be invoked.

Working directory:* /u/jazz40

The working directory to be used by the invoked command process.

Connection time-out: 0

Connection time-out in seconds.

Starting the ISPF gateway from pre- or post-deployment commands

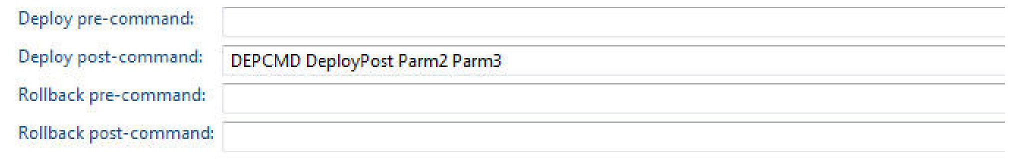
The running of executable files from deployment differs from build and promotion in that it starts the ISPF gateway directly by using the `startispf.sh` shell command. The executable files passed through the pre- and post-deployment commands must be in a certain format. This is true for all of the deployment commands:

- Load pre-command
- Load post-command
- Deploy pre-command
- Deploy post-command
- Rollback pre-command
- Rollback post-command

The following rules apply when calling deployment commands:

- The command to be run must exist in a library that is allocated to SYSPROC, SYSEXEC, or ISPLLIB in the ISPF.conf that deployment uses.
- You can pass up to 10 parameters to the command being called.
- You cannot use double quotation marks in the command.
- Single quotes in the original command are removed.
- The parameters are enclosed in double quotation marks by the `startispf.sh`.
- The REXX executable file must strip the double quotation marks that are passed through in the arguments.

The following example shows how to run executable files from a post-deployment command:



Deploy pre-command:

Deploy post-command: DEPCMD DeployPost Parm2 Parm3

Rollback pre-command:

Rollback post-command:

The following example shows how the REXX executable file must parse the argument passed to it:

```
/* REXX */
  Arg Parms
  Parse var Parms '"'Parm1'" "'Parm2'" "'Parm3'"' .
```

The example code accepts three parameters, and the final period (.) discards from the `startispf.sh` script the rest of the double quotation marks passed.

Starting the ISPF gateway from a build.xml Ant script

Another method for using `BLZGTWY` is to call the ISPF gateway directly from Ant for Enterprise Extensions xml scripts. For example,you can create a build.xml file to build your zComponent projects. The following example shows how to create an Ant xml target using `BLZGTWY` to run another REXX executable file:

```
<!-- Macro definition for ZIP command -->
  <macrodef name="zipISPF">
    <sequential>
      <exec executable="tso" failonerror="true">
        <arg line=""EXEC 'MYHLQ.SBLZSAMP(BLZGTWY)'
            'TSO EX MYHLQ.EXEC(ZIPEXEC) parm1 parm2 parm3'""/>
      </exec>
    </sequential>
  </macrodef>

<!-- ZIP -->
<target name="zip" description="zip ISPF">
  <startBuildActivity label="zip ISPF">
   autoComplete="true"
   buildResultUUID="${buildResultUUID}"
   repositoryAddress="${repositoryAddress}"
   userId="${userId}"
   passwordFile="${passwordFile}"/>
 <zipISPF/>
</target>
```

Starting the ISPF gateway from TSO for testing purposes

The following example starts the ZIPEXEC file through the ISPF gateway and passes three parameters. You can test the ISPF gateway invocation from ISPF option 6. For example, to run the TESTEXEC executable file through the ISPF gateway with logging turned on, use the following command:

```
EXEC 'MYHLQ.SBLZSAMP(BLZGTWY)' 'ISPF EX MYHLQ.EXEC(TESTEXEC) LOG=YES'
```

To run the **`TSO TIME`** command through the ISPF gateway, use one of the following commands:

```
EXEC 'MYHLQ.SBLZSAMP(BLZGTWY)' 'TSO TIME'
```

or

```
EXEC 'MYHLQ.SBLZSAMP(BLZGTWY)' 'ISPF TIME'
```

The TSO or ISPF parameters that are passed indicate to the ISPF gateway what to allocate, which is either the required TSO allocations or the full ISPF allocations.

Troubleshooting the ISPF gateway configuration

Verify that the ISPF gateway is properly configured. Any user IDs that run Rational Team Concert functions that use the ISPF gateway must have the required permissions that are described in ISPF Planning and Customizing (GC34-4814) at http://publibfp.dhe.ibm.com/epubs/pdf/ispzpc80.pdf.

The Rational Team Concert `startispf.sh` script uses UNIX System Services commands that need access to the `/tmp` directory. The build user ID that starts the `startispf.sh` script must have access to the `/tmp` directory. You can override the use of `/tmp` as the temporary directory by adding the following line to `/etc/jazz406/ccm/startispf.sh`:

```
export TMPDIR=/yourTemporaryDir
```

where *yourTemporaryDir* is a directory with write access for build users.

Installing and configuring the Rational Build Agent on a z/OS system

This section describes how to complete the installation of the Rational Build Agent a z/OS system.

The build agent executable file has been installed into `@pathPrefix@/usr/lpp/jazz/v4.0.6/buildagent` during the SMP/E installation. You must install FMID HRBA406 as part of the SMP/E package installation of Rational Team Concert prior to completing these steps.

Refer to "Installing and configuring Jazz Team Server and the Rational solution for Collaborative Lifecycle Management applications on z/OS systems" on page 15 for information about the FMIDs that can be installed independently from each other. The SMP/E package is contained in a .zip file, along with the program directories. The program directories are also available on the web at http://www-01.ibm.com/support/docview.wss?&uid=swg27041020.

Creating additional Rational Build Agent directories

On z/OS, the Rational Build Agent requires several directories in addition to the z/OS UNIX System Services directory created by the SMP/E installation.

One of the additional directories is a working directory and the other directories are used to store configuration files. There is a sample member for each application (`BLZCPBFA`) in *`hlq`*`.SBLZSAMP` used to create and populate these directories with the required files from the SMP/E installed directory.

The following three symbolic names are used throughout this guide to indicate the following directories:

Table 8. Symbolic names for directories

Symbolic name	Use	Variable in BLZCP* jobs	Default directory
@pathPrefix@	The path prefix specified during the SMP/E installation.	BLZHOME	
@confPath@	The Jazz Team Server configuration files directory.	BLZCONF	`/etc/jazz406`
@workPath@	The Jazz Team Server working directory.	BLZWORK	`/u/jazz406`

Note: The Jazz Team Server and CCM application directories require RACF file access permissions. For details, see "Security on z/OS systems" on page 5.

In addition to creating and populating these directories, the sample jobs provided also perform several required customizations. The variables mentioned in the symbolic name table, must be set in several places in each job. In addition there are other variables that might need to be set that are listed in the following table:

Table 9. Variable names and usages

Variable Name	Use	Default value
BLZBFAH	The path prefix of the Buildforge agent directory /buildagent	`/usr/lpp/jazz/v4.0.6/buildagent`
BLZBFAC	The Build Forge agent configuration files directory	`/etc/jazz406/ccm`
iconvLoc	The location of the iconv utility	`/bin/iconv`

The instructions contained in each configuration job will indicate which variables you must configure. The other application configuration jobs are optional based on which server components you plan to use.

Configure member `BLZCPBFA` in *`hlq`*`.SBLZSAMP` using the instructions contained in the member. Use the SUBMIT command to submit the modified JCL and check the job log. Return codes of 0 indicate the configuration is correct.

Attention: The `BLZCP*` members contain the *@confgrp@* and *@workgrp@* variables, which must be set to the SAF groups that contain all of the user IDs that require write access to the configuration and work directories. You must also associate group IDs (GID) with these SAF groups.

Configuring the Rational Build Agent shell script

The Jazz Build System contains a sample script named `startbfa.sh` that you can use to start the Rational Build Agent.

You can find the sample script in the `@pathPrefix@/usr/lpp/jazz/v4.0.6/buildsystem/buildtoolkit/examples/startbfa` directory, where *`@pathPrefix@`* is any prefix you specify when you install the SMP/E. This script is required to start the Rational Build Agent to specify credentials so that the agent can connect to the Jazz Team Server, and to provide you with access to libraries you need for Ant with Enterprise Extensions and native compilation. This script is copied to the Rational Team Concert configuration directory, typically `/etc/jazz406/ccm` by the BLZCPBTK job.

Important: The script must be configured to suit your environment. Many of the variables required will have been configured by the BLZCPBTK copy and configuration job. However, some of the variables, such as user ID and password information will not be set automatically. Review and replace the required variable strings in the sample script as described in the following table:

Table 10. Sample script variable strings

Variable	Description
@pathPrefix@	Directory path to prefix the Jazz directory **Note:** This is the prefix to the Jazz directory, so your prefix should include any prefix you specified as part of the SMP/E installation, and then `/usr/lpp`.
@javaPathPrefix@	Directory path to the IBM 64-bit SDK for z/OS Java 2 Technology.
@yourUserid@	The Jazz user ID.
@yourPasswordFile@	The Jazz password file as defined by the BLZBPASS job. See Running the Jazz Build Engine on z/OS at http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0m6/topic/com.ibm.team.build.doc/topics/t_rtcz_runbldeng.html in the Rational solution for Collaborative Lifecycle Management information center for information on how to run the `BLZBPASS` job.
@stepLib@	The STEPLIB DD (data set definition) to use in your z/OS build. For example, `SYS1.LINKLIB:CEE.SCEERUN`
@bfaBinPath@	Directory path to the Rational Build Agent executable directory.
@bfaConfPath@	Directory path to the Rational Build Agent configuration file directory.
@zLang@	Encoding used in host files. For example, IBM-037.
@timeout@	Timeout value in seconds for build step execution. Default if not specified is 300 seconds.
@workPath@	Directory path where metadata are stored for PDSEs. By default this is `/u/jazz406/ccm/`.
@yourHomeDirectory@	Your home directory. Use this only when you have not defined the *HOME* environment variable somewhere else.
_CMDSERV_BASE_HOME	Indicates the location of the ISPF-supplied files so the Rational Build Agent can start the ISPF gateway. See “Configuring the ISPF gateway for build, deployment, and promotion support” on page 46 for more information on *_CMDSERV_BASE_HOME*.

When you are ready to start the Rational Build Agent and accept build requests you can run the startbfa.sh script one of the following ways:

- Start the startbfa.sh script manually from OMVS
- Configure the sample started task BLZBFA supplied in the `BLZ.SBLZSAMP` library by following the instructions in the job. You will then be able to run the Build Forge Agent as a started task at system startup. You must copy this job to a system allocated PROCLIB and set up required security settings in order for it to run as a started task.

Note: Unless another shell is specified in your `bfagent.conf` file, the default shell (`/bin/sh`) is used in the session started by the Rational Build Agent on z/OS. The default shell on z/OS always runs the `/etc/profile` script for setting system-wide configurations after the `startbfa.sh` runs and it is possible that some environment variables set in the `startbfa.sh` shell script can be overridden.

When you run an Ant with Enterprise Extensions build or dependency build with either the `–verbose` or `–debug` option in the Ant arguments, the UNIX export command runs before Ant is invoked. You can look at the outputs from the export command in the build log to make sure the environment variables are correctly set.

If you need to specify other environment variables to control how the build agent runs, you can either add UNIX export commands to the startbfa.sh script, or ensure that the environment variables are previously set for the user environment where the build runs. For example, adding:

```
export _BPXK_SETIBMOPT_TRANSPORT=tcp_stack_name
```

to the startbfa.sh shell script uses the USS *_BPXK_SETIBMOPT_TRANSPORT* variable to specify the name of the TCPIP stack used.

If some environment variables are overridden by `/etc/profile`, ask your system programmer to change `/etc/profile`. If it is not possible to change `/etc/profile`, you can use the bash shell, which is available with the z/OS Ported Tools at http://www.ibm.com/systems/z/os/zos/features/unix/bpxa1ty1.html. Specify the `–noprofile` option by adding following two lines to your `bfagent.conf` file:

```
shell /bin/bash
shellflag --noprofile
```

Completing the installation and running the Rational Build Agent on a z/OS system

This section describes how to start running the Rational Build Agent on a z/OS system.

The agent runs as a standalone daemon and uses the default agent port 5555. To change the default port, use the port setting in `bfagent.conf`. See the section “Configuring the Rational Build Agent on a z/OS system” on page 58 for information about how to locate the `bfagent.conf` file and change the default port.

Complete these steps to finish the installation and to start the Rational Build Agent:

1. Start the build agent by one of these options:
   - **Option 1:** If you plan to use Ant with Enterprise Extensions builds, the Rational Build Agent must be started from a shell script that has additional environment variables set. For z/OS, the Build System Toolkit contains a sample shell script that configures those environment variables and starts the agent. For more information, see “Configuring the Rational Build Agent shell script” on page 55.

     To start the Rational Build Agent, configure the sample started task `BLZBFA` supplied in the `*hlq*.SBLZSAMP` library by following the instructions in the job. You can run the Rational Build Agent as a started task at system startup. You must copy the job to a system allocated PROCLIB and set up required security settings in order for it to run as a started task.

Note: If you include the `PORT` or `PORTRANGE` statement in `PROFILE.TCPIP` to reserve the Rational Build Agent ports, many binds are accomplished by threads in a thread pool. The job name of the Rational Build Agent thread pool is BLZBFA*x*, where BLZBFA is the name of the started task and *x* is a random single digit number. Because of the random digit, wildcards are required in the definition:

```
5666  TCP BLZBFA* ; Rational Build Agent
```

Alternatively, you can start the `startbfa.sh` script manually from OMVS.

Note: Starting the `startbfa.sh` directly as a shell script from the Rational Developer for System z UNIX shells can cause conflicts because the ISPF gateway will pick up configuration files from the Rational Developer for System z `rsed.envvars`. To avoid potential conflicts, either start the Rational Build Agent as a started task, through Internet daemon (InetD), or configure it so it is invoked directly in OMVS on z/OS and not through Rational Developer for System z.

- **Option 2:** Start the build agent directly. This method does not set up the environment variables required by Ant with Enterprise Extensions. Change directories to `@pathPrefix@/usr/lpp/jazz/v4.0.6/buildagent`, and use the -s option:

```
/usr/lpp/jazz/v4.0.6/buildagent/bfagent -s -f /etc/jazz406/ccm/bfagent.conf
```

- **Option 3:** Start the build agent through InetD.

2. To test the build agent running on the z/OS system, use the telnet command to test the connection. For more information, see Testing the connection at http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0m6/topic/com.ibm.jazz.install.doc/topics/RTCz_agent_troubleshooting_test_connection.html in the Rational solution for Collaborative Lifecycle Management information center.

Note: The build agent typically uses administrative privileges such as root or admin to log on to the operating system. Additionally, the build agent runs all commands using the permissions of the user who started the agent, not the user name used to log in. If the build agent is not run as the root or admin user, then you might receive authentication errors when testing the connection. To run the agent from a non-root or non-admin user ID, configure the magic_login setting in the bfagent.conf. This is an alternative to standard system authentication. With this setting, the system can authenticate your login with a single user name and password. To see the steps required to configure the magic_login, see bfagent.conf Reference.

Configuring the Rational Build Agent on a z/OS system

This section describes how to configure the agent after installation

Locating the agent configuration file for z/OS systems

The agent configuration file, `bfagent.conf`, provides runtime configuration of the agent's operation.

For a detailed explanation of all the options, see “bfagent.conf file for the Rational Build Agent on z/OS systems” on page 60.

The build agent configuration file, `bfagent.conf` is installed into `@pathPrefix@/usr/lpp/jazz/v4.0.6/buildagent` during the SMP/E installation. For the file to be modified, it must be copied to the configuration directory, typically `/etc/jazz406/ccm`, by the `BLZCPBFA` job.

You can specify an alternate configuration file. The `bfagent.conf` file that is used will come from a configuration directory. Therefore, the `-f` command line option is specified when the agent is started. For example:

```
bfagent -f /etc/jazz406/ccm/bfagent.conf
```

Note: The agent is generally started by using the startbfa.sh shell script. For more information, see "Completing the installation and running the Rational Build Agent on a z/OS system" on page 57..

Note: Latest entries in the jazz.net library on installing the Rational solution for CLM products on IBM i

Changing the Rational Build Agent port on z/OS systems

You can change the server port after the agent is installed.

If you install the agent on a server where port 5555 is already occupied, you can change the agent port after it is installed.

To change the port:

1. Modify the port value in your `bfagent.conf` file.
2. If you are running a stand-alone server (that is, you started with the `bfagent -s` or `bfagent -f` command), you can use the `kill` command to stop the current bfagent process and restart using the command you used to start the agent the first time.
3. If you are running using **`inetd`**, you also must modify your port setting in the `/etc/services` file and restart **`inetd`**.
4. If you are running the agent as a started task, you must stop and restart the started task.

Configuring a different shell for the Rational Build Agent on z/OS systems

You can configure an agent to use a shell other than the default shell by editing parameters in the `bfagent.conf` file.

For example, to use the tcsh shell, you can set the shell parameter as follows:

```
shell /bin/tcsh
```

For more information on the `bfagent.conf` file, see "bfagent.conf file for the Rational Build Agent on z/OS systems" on page 60.

bfagent executable file for the Rational Build Agent on z/OS systems

The bfagent executable file starts the Rational Build Agent. It reads its configuration from a `bfagent.conf` file in the same directory.

On z/OS, the agent is typically started using the `startbfa.sh` script.

The syntax for the command is:

```
bfagent [-f configfile | -s]
```

Options:

-f configfile
: Run using the configuration file in `configfile`, rather than `bfagent.conf`. This is a runtime option on IBM i, z/OS, UNIX, or Linux. It is a debugging option when running the agent manually on Windows. It cannot be used for starting the service on Windows.

-s
: Start as a standalone service. You can use this option only on IBM i, z/OS, UNIX, or Linux. Running this way is an alternative to starting bfagent with either the **`inetd`** or **`xinetd`**.

The bfagent executable file supports secure communications using z/OS System SSL.

There is another bfagent executable file provided named bfagent-openssl that supports secure communications using OpenSSL, which has been supported in previous releases. You should use the bfagent executable file whenever possible, and use the bfagent-openssl executable file only when you need to use OpenSSL. Both executable files support the same options.

Important: With the availability of z/OS System SSL, support for secure communications using OpenSSL is no longer a requirement on z/OS. The OpenSSL support for secure communications on z/OS is **deprecated** in Rational Team Concert versions 4.0.5 and higher. Instead, use z/OS System SSL.

bfagent.conf file for the Rational Build Agent on z/OS systems

The `bfagent.conf` file stores settings for how the Rational Build Agent runs. The file is in the same directory as the bfagent executable file.

The file lists all settings and internal defaults. Inactive settings are commented out.

Settings:

activity_log *path*
: Turns on activity logging. The information is appended to the file specified by *path*. The path must exist and the agent user must have write permission for it.

  Note: The agent does not report an error if the path does not exist or if it cannot write to the file.

  Important: There is no limit on file size. The file must be manually deleted. This setting is intended to be used temporarily for debugging the agent. It is not intended as a permanent log for a working agent.

allow IP-address-or-range [...]
: Use this setting for these conditions only: :

  - Agents running on Windows
  - Agents running in standalone mode on UNIX or Linux when the -s option on startup is used.

  This setting limits connections to the agent. Connections are allowed only from the IP addresses that match IP-address-or-range. By default connections are allowed from all addresses.

Specify one or both of the items:

- **IP Address**: A fully-qualified IPv4 or IPv6 address. For example, for IPv4, 255.192.192.003. The specific IP address is allowed.
- **IP Address range**: A partially-qualified IPv4 or IPv6 address. These examples are correct for IPv4, 192.168 or 192.168.63. All IP addresses that match this qualification are allowed.

Note: If you are running the agent on a superserver such as **`inetd`** or **`xinetd`**, use another method to control access. You can use a firewall, TCP wrappers (hosts.allow and hosts.deny), or the built-in filtering capability of **`xinetd`**.

bind
This setting allows the user to specify an explicit bind address for the agent. This, together with the "port" setting, determines how the agent will listen for connections when it is started with the -s command line option. The value given in the `bfagent.conf` file will force the agent to bind to the IPv4 localhost address; thus, the agent will only receive connections from a console that is on the same computer. Example: `bind 255.192.192.003`

Note: It has no effect on Windows or UNIX agents that are started by the system's service architecture, such as **`inetd`**, **`xinetd`**, or **`launchd`**.

ccviewroot root-path
This setting specifies the default view root for this host. See ClearCase® documentation on init for more information. The internal defaults are as follows:

- Windows: ccviewroot M:
- UNIX or Linux: ccviewroot /view

cc_suppress_server_root
If set, then the view path is the path set by ccviewroot. If not set, then the path set in the server definition is appended to the path set by ccviewroot. This setting does not need a value. If it is present in bfagent.conf, then it is set.

command_output_cache size
This setting causes the agent to cache output until it reaches the specified size in bytes. The internal default is not to cache. Using a cache can significantly improve agent performance and reduce network overhead. The cache size depends on how much output that command produces.

Minimum value: 2048. A value of 2048 is used internally if the setting is less than that.

cygwin

This setting is used only with agents on Windows.

This setting enables the agent to work on a Windows host using Cygwin, a Linux-like environment. When using Cygwin, a number of Linux tools are available to the agent.

When you use this setting, you might need to set cygwin_script_magic and shell settings also. The example shows one way to configure these settings:

```
cygwin
shell C:\cygwin\bin\bash.exe --login -c "%s"
cygwin_script_magic #!/bin/bash
```

The shell setting must match your installation of Cygwin.

cygwin_script_magic

This setting is used only with agents on Windows when **cygwin** is set.

This setting specifies the `#!` line to use when executing steps. The default is `#!/bin/bash`.

default_logon_domain
: Specifies the domain to use when an authentication request does not include a domain. If not specified, the agent machine's domain is used.

digest_algorithm SHA2
: When password encryption is enabled, the bfagent uses `SHA1` by default. To use SHA2, enable this setting. This setting is new in version 8.0.

When the bfagent upgrades to 8.0, this setting is not automatically added to bfagent.conf. You must add this setting to bfagent.conf when you want to use `SHA2`. If you are installing the bfagent 8.0 directly, bfagent.conf has this setting.

If you have already encrypted the password, after enabling this setting, you must encrypt the password again.

To use `SHA2`, make sure that the Digest Algorithm system configuration setting on the Build Forge® console is set to `SHA2`.

Note: If you want to use whole Secure Hash Algorithm 2 (SHA2) and enable password encryption in both the Build Forge management console and the agent, the password encryption configuration properties file bfpwcrypt.conf might need to be updated after you change to `SHA2` from `SHA1`.

disable_telnet_support
: For best results, use telnet to test the agent connection.

For the agent, there is some built-in processing overhead associated with processing and correctly handling telnet control sequences.

Use this setting to disable the agent from handling special telnet character codes which can slightly improve performance. In product environments, use this setting to benefit from the improved performance.

disable_transcode
: Turns off processing that the agent performs to convert international data when the operating system is not using UTF-8 encoding. To avoid mixed encodings and data corruption, use UTF-8 for the agent operating system.

If the operating system does not use UTF-8 encoding, the agent must convert data to the correct encoding for the operating system's locale settings.

If your operating does not use UTF-8, use this setting for best results and improved performance of the agent.

enable_agent_dll
: This setting enables DLL process tracing, which is a debugging tool.

env_recursion_limit number-of-recursions
: Sets the variable-replacement recursion limit for pre-parsing. If not set, the limit is 32.

extensions
: This setting specifies paths to external libraries of functions. The functions can be used as dot commands in a step. If this setting is not specified, external libraries are not loaded.

During parsing, the first token in the step command is taken as the function name. The second token is a string, and the third is an integer timeout value (in seconds).

Requirement: Dynamic loader support in the operating system. For example, in UNIX or Linux you need `/usr/include/dlfcn.h`. These defaults values are used internally.

- UNIX or Linux: `/usr/local/bin/bfextensions.so`
- Windows: `c:\program files\ibm\build forge\agent\bfextensions.dll`

getaddrinfo_using_addrconfig
This setting is used only for running the agent as a standalone service on UNIX or Linux operating systems (bfagent -s). This setting makes the agent use AI_ADDRCONFIG when calling getaddrinfo() to select a listening interface. By default AI_ADDRCONFIG is not used.

If you use this setting, the agent ignores interfaces that do not have a properly configured address. It listens only for interfaces that have a properly configured address.

gsk_ssl_key_location [<*kdb_path*> | <*SAF_specification*>]
Specifies either a full path to a kdb file or a SAF key ring specification.

gsk_ssl_kdb_password <*password*>
Password for the kdb file. It can be in plain text or encrypted text. Use `NULL` if a SAF key ring is used. Use **`bfagent -e`** <*plaintext*> to create the encrypted password from plain text.

gsk_ssl_protocol <*protocol*>
The protocol to use, one of `ALL` (the default), `SSLV2`, `SSLV3`, `TLSV1`, or `TLSV1_1`.

gsk_ssl_cipher_v2 <*seed*>
The cipher suite to use for system SSL version 2 (`SSLV2`). The default value is **`6321`**, which should serve most applications. See the System z documentation for more information.

gsk_ssl_cipher_v3 <*seed*>
The cipher suite to use for system SSL version 3 (`SSLV3`). The default value is **`0906030201`**, which should serve most applications. See the System z documentation for more information.

gsk_keyring_label <*label*>
The key label in the kdb file.

gsk_password_encrypt [true | false]
Used to refer to an encrypted password. If set to `true`, use **`bfagent -e`** <*plaintext*> to create an encrypted value and set **`gsk_ssl_kdb_password`**. It is set to `false` by default.

gsk_ssl_client_authentication [true | false]
Specifies whether to validate the client certificate. The default is `false`.

lang *lang-code*

This setting specifies the language that the agent uses to write messages and command output. Typically it is not set explicitly because the agent uses the language that the Management Console specifies. However, setting the language can be useful if the desired locale is not available on the computer. The setting is also useful as a backup, in case the Management Console fails to communicate a language or communicates an invalid language.

The internal default is `en`, as if it were explicitly set as follows:

```
lang en
```

leave_tmp_file

Use this setting only while you are troubleshooting.

This setting causes the temporary file that is used to hold step commands to be retained, rather than deleted after command execution. In troubleshooting, the file can be compared to the steps as they are displayed in the Management Console.

Note: Do not use this setting for typical operations.

locale locale-code.charset-code

This setting is used only with UNIX and Linux operating systems. Windows handles locales differently.

This setting specifies the language and multibyte character set that localized applications use. This setting works by setting the LANG environment variable for the agent context.

To set up the agent to treat command output as US English UTF-8, use the UTF-8 locale for your operating system. For example, on Linux use the following representation.

```
locale en_US.UTF-8
```

To determine the correct representation of the UTF-8 locale for your operating system, run the **locale -a** command.

If this setting is not specified, the agent uses the locale of the operating system. This setting is a convenience. This setting is especially useful if the default locale of the operating system is not the locale that you want the agent to use. The setting is especially useful if changing the system locale to meet agent requirements is not practical.

magic_login user:encoded-password

The agent typically uses administrative privileges such as root or admin to log on to the operating system. The magic_login setting is an alternative to standard system authentication. With this setting, the system can authenticate your login with a single user name and password.

If the agent is run as the root or admin user, this setting is ignored and normal authentication is attempted.

The agent runs all commands using the permissions of the user who started the agent, not the user name used to log in.

This setting is used in only these situations:

- When running the agent with administrative privileges is not possible. For example, use this setting with UNIX systems that do not work with PAM.
- When running the agent with administrative privileges is not permissible because of security policies.

To configure a login for the agent:

1. Create a server authentication that uses a user name and password. In the Management Console, click **Servers > Server Auth**.
2. For this example the user name is `build` and the password is `MySecretPassword`.

3. Create a server that uses the agent. Associate the server authentication with this server in the **Authentication** field.
4. Generate an encoded password for the agent. In the installation directory for the agent, run **bfagent -P** with the password that you choose.

   An SMD5 hash-encoded password is returned, as follows:

   ```
   bfagent -P MySecretPassword
   eca0b7f2f4fbf110f7df570c70df844e1658744a4871934a
   ```
5. In `bfagent.conf`, set magic_login to use the correct user name and encoded password.

   ```
   magic_login build:eca0b7f2f4fbf110f7df570c70df844e1658744a4871934a
   ```
6. Start the agent.
7. Test the server connection. In **Servers**, select the server, and then click **Test Server**.

map *drive-and-user-spec***[; ...]**
This setting specifies a mapped drive. Some systems might require drive mappings. For example, a drive mapping might be required because a shell is run from a shared drive. Mappings specified on the agent are performed before mappings specified by _MAP environment variables in the Management Console. This example illustrates two drive mappings:

```
map X:=//host1/share;Z:=//host2/share(username,password)
```

map_drive_is_failure
When specified, this setting causes a step to fail upon encountering an unmapped drive specification, before step execution. If this setting is not specified, steps ignore drive failures and attempt to run the step. In that case ensure that the failure generates a meaningful error message.

no_preparse_command
This setting disables the variable-expansion parsing that the agent typically performs on a command before passing the command to the shell. See also the _NO_PREPARSE_COMMAND environment variable, which can be used for an single project or step.

no_pty

This setting is used only with agents that are running on UNIX or Linux systems.

This setting can be used to help prevent the system shell from locking up when the shell interacts with the pseudoterminal of the agent. This setting is typically used with HP/UX and z/OS. You can also use two other methods to help prevent this kind of lockup:

- Use an alternate shell.
- Use the nologonshell setting

The **no_pty** setting disables the pseudoterminal allocation.

Note: Using **no_pty** affects some commands. For example, the **ls** command returns output in a single column rather than three columns. If you use this setting, test thoroughly before you deploy the job to a production environment.

nologonshell

Use this setting only with agents that are running on UNIX or Linux.

This setting causes the shell that the agent runs to be a normal shell, not a logon shell. This setting is often in these cases:

- The logon shells provide verbose output.
- The logon shells change environment settings in unwanted ways.
- The logon shells attempt to communicate interactively with the user.

When set, standard methods of requesting that the shell be a normal shell rather than a logon shell are used. This might not work on all platforms and in such cases, the *shellflag* setting might be used to pass flags to the shell in order to modify its behavior.

Those behaviors are not desirable for the agent, because it runs as a user without being an interactive user.

Note: The Mac OS X 10.5 system uses `/bin/bash`, which does not respond to nologonshell. Use `shellflag -l`.

Note: The z/OS operating system always uses the `/etc/profile` script for both logon shells and non-logon shells. You might need to change the contents of the script or use another shell if its behavior does not work well with the agent.

See also the **shellflag** setting. Flags can be used to change logon script behavior.

password_encrypt_module *dll_path*;*conf_path*
Required to enable SSL on the agent. It specifies paths to a DLL and configuration file.

- *dll_path* is the path to `bfcrypt.dll` (it is typically `./bfcyrpt.dll`).
- *conf_path* is the path to `bfpwcrypt.conf` (it is typically `./bfcrypt.conf`).

port *port-number-or-range* **[...]**

This setting is used only with agents that are running in standalone mode on UNIX or Linux when you issue the **-s** option on startup.

This setting specifies the port that the agent uses to listen for connections with the Management Console.

Specifies the port that the agent uses to listen for connections with the Management Console.

Note: The port is set to 5555 by default. For UNIX or Linux the setting is in `/etc/services`.

read_timeout

Time in seconds that the agent waits for a request until it disconnects. The default is 1800 seconds (30 minutes). Set the value to 0 to disable the timeout.

The directive helps prevent client connection contacts from holding the port open if a legitimate engine request is not received. Some network port-scanning software behaves this way.

Do not set very small values for this directive. Normal engine behavior might include gaps between requests of several minutes.

shell *shell_name* **[***options***]**
This setting specifies the default shell. Internal defaults are as follows:

- Windows: `shell cmd.exe /q /c "%s"` unless the following settings are used:
  - If the `cygwin` setting is used, the default is `shell C:\cygwin\bin\bash.exe --login -c "%s"`
  - If the `cygwin` setting is not used, the default is `shell cmd.exe /u /q /c "%s"`
- UNIX or Linux: The shell set for the user account, or `/bin/sh` if the user's shell can not be determined. Note that you cannot specify parameters in this setting, but you can use the `shellflag` setting to pass them. The agent automatically forces the default to be a logon shell by inserting a hyphen. For example, `/bin/ksh` is sent as `-ksh`. If `shell` is set explicitly, then `nologonshell` is set implicitly. See `nologonshell`.
- *System i*®: Set the shell value to /bin/sh

You can override this setting from within a step. A step that starts with a line containing `#!` overrides the shell setting and the `nologonshell` setting is used to run the step commands.

shell_compatible_undef_vars
This setting forces the representation of undefined variables to be an empty string. If not set, the representation is the variable name for variables of format $VAR, ${VAR}, or %VAR% and the empty string for $[VAR].

shellarg

This setting is used only with agents that are running on UNIX or Linux.

Use this setting if it seems that commands are being scrambled. Some shells on Red Hat Linux Enterprise require this setting.

The setting changes the way a command script is passed to the shell. Normally the script is passed through standard input:

```
/bin/sh < /tmp/bfshellscript.sh
```

This setting causes scripts to be run by passing them as parameters:

```
/bin/sh /tmp/bfshellscript.sh
```

shellflag *flag*

This setting is used only with agents that are running on UNIX or Linux.

This setting adds a flag when a shell is running. Only one flag can be specified. It is typically used to disable **rc** script processing in order to reduce output or unnecessary processing.Examples:

- csh and derivatives: use `shellflag -f` to disable rc script processing.
- bash: use `shellflag —-noprofile` to disable profile script processing.

ssl_ca_location *path*
Specifies the keystore file that contains the certificate authority. If the agent runs as a service, use an absolute path.

ssl_cert_location *path*
Specifies the keystore that contains the private certificate. If the agent runs as a service, use an absolute path.

ssl_client_authentication [true | false]
Set to true to require client authentication when a connection is made to the agent. If true, the Build Forge engine's certificate must be added to the agent's certificate authority keystore.

ssl_cipher_group [*grouplist* | **ALL]**
: Specifies individual cipher groups to use. Can be set to ALL.

ssl_cipher_override *cyphers*
: Overrides the cipher group. Specify the ciphers to use.

ssl_key_location *path*
: Specifies the keystore file that contains the key. If the agent runs as a service, use an absolute path.

ssl_key_password *password*
: Password for the key. This property is stored in clear text by default. You can configure the agent to encrypt this password using its own key or the Build Forge server's key.

ssl_protocol *protocol*
: The SSL handshake protocol to use, one of TLSv1, TLSv1.1, TLSv1.2. The protocol must match the protocol used by the Build Forge server. When enabled, TLSv1, TLSv1.1, or TLSv1.2, it only accepts the corresponding connection. For example, when TLSv1.2 is enabled, the bfagent only accepts a TLSv1.2 connection.

  Note: IBM i does not support TLSv1/2.

update_path *path*

: This setting identifies the full path to the Build Forge agent executable. The setting is established automatically during installation. The directory is a default directory for the operating system or the installation directory that you specify.

  Note: This setting is ignored on Windows agents. The update path is taken from registry keys. The keys are set during agent installation.

win_reexec_after_auth
: Add this setting if you need to run agent commands on a network share file system using Build Forge server authentication credentials. For example, to modify files in a ClearCase dynamic view, the agent must access ClearCase files on a networked shared file system.

  The Build Forge agent initially starts up with Windows system account credentials. To run commands, the agent later authenticates with Windows using Build Forge server authentication credentials.

  Without this setting, the network share recognizes only the initial Windows system account credentials and ignores the subsequent server authentication credentials that are needed to access and write to files on the network share file system.

  The win_reexec_after_auth starts a new process after authenticating with Windows again by using the server authentication credentials and forces the shared file system to recognize the changed credentials.

  When you use the win_reexec_after_auth setting, the agent runs as a service and does not distinguish between commands that access network share files and those that do not, so you might notice a performance impact.

xstream_allow_ssl_mismatch
: Required if file transfer is needed between an agent compiled with OpenSSL and an agent compiled without OpenSSL. By default agents

compiled with OpenSSL require AES_CBC encrypted file transfers. They reject any file transfers requested using PLAIN or PRNG encoding unless this setting is used.

xstream_bind *ip_address*
: Specifies an IP address to be used only for direct file transfers. The address must be accessible by the agents that receive the files. By default an agent listens on all network interfaces. See also **bind**.

xstream_conn_timeout *seconds*
: Time in seconds that an agent waits for a connection. The engine must forward the connection request to the receiving agent and the receiving agent must establish a connection with the sending agent within this time. By default it is set to 20 seconds.

xstream_listen_range *port-range*
: The range of ports an agent listens on for connections. This setting is useful when there is a firewall between the connecting hosts. The firewall administer can configure the firewall to allow the ports permitted for connections, for example 22880-22889. The default port-range is 16384-32767. However, if *xstream_bind* is used and *xstream_listen_randomize* is not used, then the agent does not specify a range and the operating system determines what ports to use.

xstream_listen_randomize
: Causes random selection of a port within *xstream_port_range*. If not specified, the agent starts checking with the lowest port number. This setting is highly recommended as a security measure.

xstream_recv_timeout *seconds*
: Time to wait for file transfer. If during any time in the file transfer this period passes without the receiving agent getting data from the sending agent, then the transfer fails and the connection is closed. The default is 20 seconds.

xstream_send_timeout *seconds*
: If password encryption has never been enabled in the configuration properties file, bfpwcrypt.conf, use the steps in this topic after you change from `SHA1` to `SHA2`.

Note: Latest entries in the jazz.net library on installing the Rational solution for CLM products on IBM i

Build Agent Lookup Service

Rational Build Agent service plug-ins provide the capability to look up the build requests that use Rational Build Agent as the build engine.

Using the Rational Team Concert client, you can create build definitions that use Rational Build Agent as the build engine. The Build Agent Lookup Service checks build requests in the repository periodically to locate requests that use Rational Build Agent as the build engine. When the build requests are found, they are handled by the internal service and send command lines to Rational Build Agent.

The Build Agent Lookup Service handles multiple build requests in each lookup period. However, if there is more than one build definition defined to use one Rational Build Agent, and a user requests builds using these build definitions at same time, the Build Agent Lookup Service only handles one build request in the lookup period. A single Rational Build Agent should only receive one build request at a time.

Configuring a lookup service task

You can configure and request a Rational Build Agent *service lookup task*, also known as a Rational Build Agent *loop task service*, in the Jazz web client.

Go to **Jazz Administration** > **Server** > **Advanced Properties** > **Rational Build Agent**. Here, you can set the following two values:

Rational Build Agent loop task service Contributor ID

This is the ID of the user running the Rational Build Agent loop task service on the server. The default user ID is *ADMIN*.

Restriction: When you request the Rational Build Agent loop task service, you **must** use the default user ID: *ADMIN*.

Note: Make sure that the user running the Rational Build Agent loop task service has permission to perform build actions. The following conditions must be met before a user can perform build actions:

- The user must be a member of the specified project and team areas.
- The user must be set to a specific role in the specified project and team areas. If you do not set a role, the default is *everyone*.
- The user role must have full permissions for build actions for the specified project and team areas. You can define the permissions here: **Project Area** > **Process Configuration** > **Team configuration** > **Permissions**.
- The user must be assigned to either a Developer or Build System Client Access License (CAL).

Rational Build Agent loop task service interval time

This value defines the interval running time, in seconds, of the Rational Build Agent loop task service. The default value is 60 seconds.

Using the Rational Build Agent and Job Monitor to run builds using JCL

This section describes how to submit a build for native z/OS artifacts and obtain results using the Job Monitor.

You must already have installed Job Monitor (FMID HRDV406) and Rational Build Agent (FMID HRBA406) as part of SMP/E installation. For additional information, see Overview of the SMP/E installation process.

Note: If you have an instance of the Rational Developer for System z Job Monitor on your system, you can use it for job monitoring.

Job Monitor customization

The Job Monitor component depends on the SMP/E installation of FMID HRDV406. You will need the assistance of a security administrator and a TCP/IP administrator to complete this customization task.

Before you begin

This customization task requires the following resources and special customization tasks:

- APF-authorized data set
- Various security software updates
- TCP/IP port for internal communication

About this task

To verify the installation and to start using Job Monitor, you must perform the following tasks. Unless otherwise indicated, all tasks are mandatory.

Procedure

1. Define an APF-authorized data set. For details, see "PARMLIB changes."
2. Create a started task procedure. For details, see "PROCLIB changes."
3. Customize the Job Monitor configuration files. For details, see "Job Monitor configuration file BLZJCNFG" on page 72.
4. Update security definitions. For details, see "Job Monitor security" on page 81.

PARMLIB changes:

Use various commands to set APF authorizations and modify PARMLIB definitions.

Refer to *MVS™ Initialization and Tuning Reference (SA22-7592)* for more information on the PARMLIB definitions listed below. Refer to *MVS System Commands (SA22-7627)* for more information on the sample console commands.

APF authorizations in PROGxx

For Job Monitor to access JES spool files, the following must be APF-authorized:

- Module `BLZJMON` in the *`hlq`*`.SBLZAUTH` load library, where *hlq* is the high-level qualifier you used during SMP/E installation.
- The Language Environment (LE) runtime libraries `(CEE.SCEERUN*)`

APF authorizations are defined in `SYS1.PARMLIB(PROGxx)`, if your site follows IBM recommendations.

You can set APF authorizations dynamically with the following console commands, where *volser* is the volume on which the data set resides if it is not SMS-managed:

- `SETPROG APF,ADD,DSN=`*`hlq`*`.SBLZAUTH,SMS`
- `SETPROG APF,ADD,DSN=CEE.SCEERUN,VOL=volser`
- `SETPROG APF,ADD,DSN=CEE.SCEERUN2,VOL=volser`

PROCLIB changes:

Customize the sample started task member hlq.SBLZSAMP(BLZJJCL), as described within the member, and copy it to SYS1.PROCLIB.

This task must be saved in a system procedure library defined to your JES subsystem. The IBM default procedure library is `SYS1.PROCLIB`.

Customize the sample started task member *hlq*.SBLZSAMP(BLZJJCL), as described within the member, and copy it to SYS1.PROCLIB. As shown in the code sample below, you have to provide the following:

- The high-level qualifier of the authorized load library, default BLZ. Replace BLZ with your high-level qualifier.
- The Job Monitor configuration file, default *hlq*.SBLZSAMP(BLZJCNFG)

```
//*
//* JES JOB MONITOR
//*
//JMON PROC PRM=, * PRM='-TV' TO START TRACING
// LEPRM='RPTOPTS(ON)',
// HLQ=BLZ,
// CFG=BLZ.SBLZSAMP(BLZJCNFG)
//*
//JMON EXEC PGM=BLZJMON,REGION=0M,TIME=NOLIMIT,
// PARM=('&LEPRM,ENVAR("_CEE_ENVFILE_S=DD:ENVVARS")/&PRM')
//STEPLIB DD DISP=SHR,DSN=&HLQ..SBLZAUTH
//ENVVARS DD DISP=SHR,DSN=&CFG
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
// PEND
//*
```

Job Monitor configuration file BLZJCNFG:

Modify the definitions in the BLZJCNFG configuration file to control how Job Monitor behaves.

Job Monitor provides JES-related services. You can control the behavior of Job Monitor with the definitions in BLZJCNFG.

Customize the sample Job Monitor configuration member *hlq*.SBLZSAMP(BLZJCNFG), as shown in the following sample. Comment lines start with a pound sign (#), when using a U.S. code page. Data lines can only have a directive and its assigned value. Comments are not allowed on the same line.

Note: The BLZJMON started task must be restarted to pick up any changes you make.

```
HOST_CODEPAGE=IBM-1047
SERV_PORT=6716
TZ=EST5EDT
#_BPXK_SETIBMOPT_TRANSPORT=TCPIP
#APPLID=FEKAPPL
#AUTHMETHOD=SAF
#CODEPAGE=UTF-8
#CONCHAR=$
#CONSOLE_NAME=JMON
#GEN_CONSOLE_NAME=OFF
#LIMIT_COMMANDS=NOLIMIT
#LIMIT_VIEW=USERID
#LISTEN_QUEUE_LENGTH=5
#MAX_DATASETS=32
#MAX_THREADS=200
#TIMEOUT=3600
#TIMEOUT_INTERVAL=1200
#TRACE_STORAGE=OFF
#SEARCHALL=OFF
#SUBMIT_TIMEOUT=30
#SUBMITMETHOD=TSO
#TSO_TEMPLATE=BLZ.SBLZSAMP(BLZTSO)
```

HOST_CODEPAGE
The host code page. The default is `IBM-1047`. Change to match your host code page.

SERV_PORT
The port number for Job Monitor host server. The default port is `6716`.

Note: Before selecting a port, verify that the port is available on your system with the TSO commands **`NETSTAT`** and **`NETSTAT PORTL`**.

TZ
Time zone selector. The default is `EST5EDT`. The default time zone is UTC +5 hours (Eastern Standard Time (EST) Eastern Daylight Savings Time (EDT)). Change this to represent your time zone. You can find additional information here: *UNIX System Services Command Reference (SA22-7802)*.

The following definitions are optional. If omitted, default values are applied as specified below:

_BPXK_SETIBMOPT_TRANSPORT=<tcpip stack name>
Specifies the name of the TCPIP stack to be used. The default is `TCPIP`. Uncomment and change to the requested TCPIP stack name, as defined in the `TCPIPJOBNAME` statement in the related TCPIP.DATA.

Note: Coding a SYSTCPD DD statement in the server JCL does not set the requested stack affinity.

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

AUTHMETHOD
The default is SAF, which means that the System Authorization Facility (SAF) security interface is used. Do not change unless directed to do so by the IBM support center

CODEPAGE
The workstation code page. The default is `UTF-8`. The workstation code page is set to `UTF-8` and generally should not be changed. You might need to uncomment the directive and change `UTF-8` to match the workstation's code page if you have difficulty with NLS characters, such as the currency symbol.

CONCHAR
Specifies the JES console command character. `CONCHAR` defaults to `CONCHAR=$` for JES2, or `CONCHAR=*` for JES3. Uncomment and change to the requested command character

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 correct console name, using the guidelines below.

- `CONSOLE_NAME` must be either a console name with two to eight alphanumeric characters, or `&SYSUID`.
- 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
```

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.

Note: The only valid settings are ON and OFF.

LIMIT_COMMANDS

Defines against which jobs the user can issue selected JES commands (Show JCL, Hold, Release, Cancel, and Purge). The default (LIMIT_COMMANDS=USERID) limits the commands to jobs owned by the user. Uncomment this directive and specify LIMITED or NOLIMIT to allow the user to issue commands against all spool files, if permitted by your security product.

Table 11. 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

Note: The only valid settings are USERID, LIMITED, and NOLIMIT.

LIMIT_VIEW

Defines what output you can view. With the default (LIMIT_VIEW=NOLIMIT) you can view all JES output, if permitted by your security product. Uncomment this directive and specify USERID to limit the view to output.

Note: The only valid settings are USERID and NOLIMIT.

LISTEN_QUEUE_LENGTH
The TCP/IP listen queue length. The default is `5`. Do not change unless directed to do so by the IBM Software Support.

MAX_DATASETS
The maximum number of spooled output data sets that Job Monitor will return to the client (for example, `SYSOUT`, `SYSPRINT`, `SYS00001`, and so on). The default is `32`. The maximum value is `2147483647`.

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

TIMEOUT
The length of time, in seconds, before a thread is killed due to lack of interaction with the client. The default is `3600` (1 hour). The maximum value is `2147483647`. `TIMEOUT=0` disables the function.

TIMEOUT_INTERVAL
The number of seconds between timeout checks. The default is `1200`. The maximum value is `2147483647`.

TRACE_STORAGE
Enable storage tracing. The default is OFF. Uncomment this directive and specify ON to write a storage report to DD SYSOUT after each command. The only valid values are ON and OFF. Use only when directed by the IBM support center.

SEARCHALL
Collect APPC and z/OS UNIX output that matches the JES Job Monitor filter, for example output written to SYSOUT by a Developer for System z CARMA server started using the CRASTART method. The default is OFF. Uncomment this directive and specify ON to collect the additional spool files. The only valid values are ON and OFF.

SUBMIT_TIMEOUT
The number of seconds that Progress Monitor will wait for the completion of the `TSO_TEMPLATE` job. The default is `30`. The maximum value is `2147483647`.

Note: `SUBMIT_TIMEOUT` has no effect unless `SUBMITMETHOD=TSO` is also specified.

SUBMITMETHOD=TSO
Submit jobs through TSO. The default (`SUBMITMETHOD=JES`) submits jobs directly into JES. Uncomment this directive and specify `TSO` to submit the job through TSO **SUBMIT** command. This method allows TSO exits to be invoked; however, it has a performance drawback and for that reason it is not recommended.

Notes:

1. The only valid settings are `TSO` and `JES`.
2. If `SUBMITMETHOD=TSO` is specified, then you must also define `TSO_TEMPLATE`.

TSO_TEMPLATE
Wrapper JCL for submitting the job through TSO. The default value is `*hlq*.SBLZSAMP(BLZTSO)`. This statement refers to the fully qualified member

name of the JCL to be used as a wrapper for the TSO submit. See the `SUBMITMETHOD` statement for more information.

Notes:

1. A sample wrapper job is provided in `BLZ.SBLZSAMP(BLZTSO)`. Refer to this member for more information on the customization needed.
2. `TSO_TEMPLATE` has no effect unless `SUBMITMETHOD=TSO` is also specified.

Submitting JCL inside the build command

Use the Rational Build Agent to submit JCL to JES. An agent running on z/OS can monitor and report build results by communicating with an instance of the Job Monitor running on the same z/OS system.

Rational Build Agent supports JCL submission through the **`.submitJCL`** command.

Prerequisites and security considerations:

To submit JCL to the Job Monitor, the Job Monitor must be running on the same system as the build agent.

When JCL is submitted using the build agent and Job Monitor, you can control what user the JCL is submitted under by whether the build agent is started by a super user or non-super user and by which configuration parameters you use in `bfagent.conf`.

The following `bfagent.conf` parameters apply to JCL submission:

- **`jcl_submit_user`**
- **`job_monitor_port`**
- **`enable_credential_retention`**

To configure the communication between the Job Monitor and the build agent, you can use either the **`jcl_submit_user`** parameter or the **`enable_credential_retention`** parameter and the**`job_monitor_port`** parameter set to appropriate values in the `bfagent.conf` configuration file.

The **`jcl_submit_user`** parameter provides a single set of credentials that will be used by Job Monitor when you submit jobs to JES. Add the following line to the `bfagent.conf` file:

```
jcl_submit_user userid:encrypted_password
```

where *userid* is the system ID of the user submitting jobs, and *encrypted_password* is an encrypted version of that user password. You can find the encrypted form of the password by running the following line at the USS command prompt:

```
bfagent –e password
```

where *password* is the password to be encrypted. The command will print a text string containing the encrypted value. The result will be similar to the following:

```
050405aaeb43166a00f763716b989f26651e2448ce309b72680a
```

The **`job_monitor_port`** parameter specifies the port with which Job Monitor is communicating. Add the following line to the `bfagent.conf` file:

```
job_monitor_port XXXX
```

where *XXXX* is the Job Monitor port. This port should match your **SERV_PORT** setting for Job Monitor, which is set to 6716 in the *hlq*.SBLZSAMP(BLZJNCFG) file, or it should match your existing Rational Developer for System z Job Monitor port.

If you want to be able to submit JCL with the credentials of different users, use the **enable_credential_retention** option instead of **jcl_submit_user**. When the **enable_credential_retention** bfagent.conf option is enabled the agent reuses the credentials used to authenticate with the agent when the agent authenticates with the Job Monitor. Individual users can submit JCL builds under their authority by using the **Build Agent Authentication Overrides** options when submitting the build. To use the **enable_credential_retention** option, the build agent must be started by a super user.

If you have both **enable_credential_retention** and **jcl_submit_user** enabled, then **jcl_submit_user** takes precedence. In either case, you can still explicitly specify the user ID and password using the -u option to .submitJCL.

For submitting JCL, the user ID and password used to authenticate with the Job Monitor will be determined in the following order:

1. -u option to .submitJCL, specified as:
   ```
   -u userid:password
   ```
2. **jcl_submit_user** data in bfagent.conf
3. Credentials used to authenticate with the agent either listed on the Build Agent tab of the build engine definition or from the **Build Agent Authentication Overrides** if **enable_credential_retention** is enabled in bfagent.conf

If the build agent is started by a super user:

- The build agent definition can use a super user or non-super user
- The priority for evaluating JCL submission is:
  1. -u user specified
  2. **jcl_submit_user**
  3. **enable_credential_retention**
- The **enable_credential_retention** will allow you to override the user using **Build Agent Authentication Overrides**

If the build agent is started by a non-super user:

- You can use -u and **jcl_submit_user**
- If you enable **magic_login** and **enable_credential_retention**, JCL will run under the magic_login user but you cannot override

Submitting JCL contained in a build system data set:

You can submit JCL contained in a data set on the target build system using the Rational Build Agent.

Job Monitor will submit the job to JES and report the results of the request. You can then view build results through the z/OS client.

1. Create a data set member containing the following JCL. Note that this job contains inline COBOL source code that will be compiled and link-edited. Customize the data set names contained in this job to values appropriate to your target system.
   ```
   //HELLO    JOB ,NOTIFY=DEARTH
   //*
   //* COBOL COMPILATION
   ```

```
//*
//COBOL    EXEC PGM=IGYCRCTL,PARM='NODECK,OBJECT,LIB'
//STEPLIB  DD DSN=COBOL.V4R1M0.SIGYCOMP,DISP=SHR
//SYSIN    DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO.
       PROCEDURE DIVISION.
       MAIN.
           DISPLAY 'HELLO, RTCZ.'.
           STOP RUN.
/*
//SYSLIN   DD DSN=DEARTH.SAMPLE.OBJ(HELLO),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT2   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT3   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT5   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT6   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT7   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*
//LINKEDIT EXEC PGM=IEWBLINK,PARM='LIST,LET,MAP,XREF,REUS,RENT'
//SYSLIN   DD *
 INCLUDE SYSLIB(HELLO)
 NAME HELLO(R)
/*
//SYSLIB   DD DSN=DEARTH.SAMPLE.OBJ,DISP=SHR
//         DD DSN=CEE.SCEELKED,DISP=SHR
//SYSLMOD  DD DSN=DEARTH.SAMPLE.LOAD(HELLO),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*
```

2. Create a build definition using the Eclipse client.
   a. In the Team Artifacts view, expand the project area folder in which you want to create a build definition.
   b. Right-click **Builds** > **New Build Definition**.
   c. Select **Create a new build**, then click **Next**.
   d. In the General Information window, enter a build definition ID and a brief description of the build definition and select **Command Line - Rational Build Agent** from the Available Templates menu. Click **Next**.
   e. In the Pre-Build window, deselect **Pre-Build Command Line** and click **Next**.
   f. In the Post-Build window, deselect **Post-Build Command Line** and click **Next**.
   g. In the Additional Configuration window, select both **General** and **Properties**, and then click **Finish**. The build definition you created opens in the Build Definition editor.
   h. On the **Overview** tab, under **Supporting Build Engine**, click **Create**.
   i. In the General Information window of the New Build Engine wizard, enter a build engine ID and a brief description of the build engine and select **Rational Build Agent** from the Available Templates menu. Click **Finish**.
   j. In the Build Definition editor, specify the following values on the **Build Command Line** tab:
      1) Enter this command line. Replace <*PDS(MEMBER)*> with the data set you created previously. Note that the command begins with a leading period.

      ```
      .submitJCL <PDS(MEMBER)>
      ```

2) Set the working directory to a fully qualified USS path on the build machine. This directory will be used as a work directory by the build process. It must exist before you can request a build.

k. Click **Save**.

3. Request a build.
   a. In the Team Artifacts view, select the build definition, right-click, and select **Request Build**.
   b. Click **Submit**.
   c. If a message like the following is displayed, click **OK** to submit the request: `The build engine does not appear to be processing requests.`
   d. In the Builds view, check the status periodically. Click **Update** to refresh the view.
4. When the build is completed, double-click the build result to view the build log.

Providing JCL through a Rational Build Agent step command:

You can specify JCL inline as part of a Rational Build Agent step command.

With this job submission method, you can use substitution parameters to specify values like the HLQ of the source data sets. The parameters will be replaced with values specified on the Build Definition properties tab prior to job submission.

1. Make sure you have data sets defined that will contain the object decks and load modules that result from COBOL compilation and link-editing.
2. Verify that you have defined a Rational Build Agent build engine.

   Note: You must complete the build engine and build engine ID steps in "Submitting JCL contained in a build system data set" on page 77 before you can verify the Rational Build Agent build engine.
3. In the build engine editor, click the **Build Agent** tab. If you set up the Rational Build Agent with secure communication in the bfagent.conf file, select **Connect securely to Build Agent**. Select the type of secure protocol for which the Rational Build Agent is configured.
   a. Enter the following information to connect to Rational Build Agent:

      `Hostname`
      Your build machine IP address or hostname.

      `Port`
      The port that communicates with Rational Build Agent. The default is port 5555.

      `User name`
      The z/OS RACF user ID of the builder on the target build machine.

      `Password`
      The z/OS RACF password.

      `Confirm Password`
      Enter again the The z/OS RACF password.
   b. Click **Test Connection**. The results of the connection test are displayed in the Rational Build Agent Connection Test Results box.
4. Create a build definition using the Eclipse client.
   a. In the Team Artifacts view, expand the project area folder in which you want to create a build definition.

b. Right-click **Builds** > **New Build Definition**.
c. Select **Create a new build**, then click **Next**.
d. In the General Information window, enter a build definition ID and a brief description of the build definition and select **Command Line - Rational Build Agent** from the Available Templates menu. Click **Next**.
e. In the Pre-Build window, deselect **Pre-Build Command Line** and click **Next**.
f. In the Post-Build window, deselect **Post-Build Command Line** and click **Next**.
g. In the Additional Configuration window, select both **General** and **Properties**, and then click **Finish**. The build definition you created opens in the Build Definition editor.
h. On the Properties tab of the new build definition, create a new property called `HLQ`. This property will be used throughout the JCL to specify the high-level qualifier to be used for source and output data sets on the target build system.
   1) Click **Add**.
   2) Select String as the property type and click **OK**.
   3) Specify `HLQ` as the name.
   4) Enter the HLQ of the target data sets and click **OK**.
i. Specify the following values on the Build Command Line tab:
   1) Enter this command line into the Command input box. Using the option `–c` with the `.submitJCL` command allows you to specify JCL as part of the command. Any occurrence of `${HLQ}` will be replaced with the value specified on the Properties tab of the build definition. Note that the command begins with a leading period. Be sure to verify that data set definition (DD) statements in the JCL contain values appropriate for your target system.

      Note: When you enter commands into the Command input box, there are some restrictions to consider:
      - Any commands you enter **must** be entered on separate lines to be recognized as separate commands.
      - When using other commands with the **`.submitJCL`** command and the -c option (**`.submitJCL -c`**), the **`.submitJCL -c`** command **must** be the last command you enter, followed by the inline JCL.

```
.submitJCL -c
//HELLO    JOB ,NOTIFY=${HLQ}
/*JOBPARM S=*
// SET HLQ=\'${HLQ}\'
//*
//* COBOL COMPILATION
//*
//COBOL    EXEC PGM=IGYCRCTL,PARM='NODECK,OBJECT,LIB'
//STEPLIB  DD DSN=COBOL.V4R1M0.SIGYCOMP,DISP=SHR
//SYSIN    DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO.
       PROCEDURE DIVISION.
       MAIN.
           DISPLAY 'HELLO, RTC.'.
           STOP RUN.
/*
//SYSLIN   DD DSN=&HLQ..SAMPLE.OBJ(HELLO),DISP=SHR
//SYSPRINT DD SYSOUT=*
```

```
//SYSUT1   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT2   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT3   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT4   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT5   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT6   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT7   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*
//* LINKEDIT
//*
//LINKEDIT EXEC
PGM=IEWBLINK,PARM='LIST,LET,MAP,XREF,REUS,RENT'
//SYSLIN   DD *
INCLUDE SYSLIB(HELLO)
NAME HELLO(R)
/*
//SYSLIB   DD DSN=&HLQ..SAMPLE.OBJ,DISP=SHR
//    DD DSN=CEE.SCEELKED,DISP=SHR
//SYSLMOD  DD DSN=&HLQ..SAMPLE.LOAD(HELLO),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*
```

2) Set the working directory to a fully qualified USS path on the build machine. This directory will be used as a work directory by the build process. It must exist before you can request a build.

j. Click **Save**.

k. Request a build:

1) In the Team Artifacts view, select the build definition, right-click, and select **Request Build**.
2) Click **Submit**.
3) If a dialog is displayed that states that the build engine does not appear to be processing requests, click **OK** to submit the request.
4) In the Builds view, check the status periodically. Click **Update** to refresh the view.

l. When the build is completed, double-click the build result to view the build log.

Job Monitor security

Job Monitor and its security mechanisms must be stored in a secure file system.

Only system administrators should be able to update Job Monitor program libraries and configuration files.

JES security:

Apply various basic access limitations to JES spool files and operator commands.

Use Job Monitor to access the JES spool. The server provides basic access limitations that you can extend with the standard spool file protection features of your security product. You must perform actions against spool files–Hold, Release, Cancel, and Purge–through an EMCS console, for which you must set up conditional permits.

Actions against jobs: target limitations:

Only some JES spool commands are available through Job Monitor, and access to these commands is also limited by file ownership.

Job Monitor does not provide full operator access to the JES spool. Only the **Hold**, **Release**, **Cancel**, and **Purge** commands are available, and by default, only for spool files that you or another user own. You can issue these commands by selecting the appropriate option in the client menu structure. There is no command prompt. You can widen the scope of the commands using security profiles to define which jobs the commands are available for.

Like the SDSF **SJ** action character, Job Monitor also supports the **Show JCL** command to retrieve the JCL that created the selected job output, and then display it in an editor. Job Monitor retrieves the JCL from JES, which can help you to find an original JCL member that is otherwise not easily located.

Table 12. 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 12 are, by default, limited to jobs you or another user own. You can change this with the **LIMIT_COMMANDS** directive, as documented in "Job Monitor configuration file BLZJCNFG" on page 72.

Table 13. Job Monitor command permission matrix

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. Like SDSF, Job Monitor also extends the use of the JESSPOOL class to protect job resources.

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

Table 14. Extended JESSPOOL profiles

Header	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 14. Extended JESSPOOL profiles (continued)

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

Use the following substitutions in the preceding table:

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

If the `JESSPOOL` class is not active, then there is different behavior defined for the `LIMITED` and `NOLIMIT` value of `LIMIT_COMMANDS`, as described in "Job Monitor configuration file BLZJCNFG" on page 72. The behavior is identical when `JESSPOOL` is active, because the class, by default, denies permission if a profile is not defined.

Actions against jobs: execution limitations:

You must have certain security authorizations to perform Job Monitor JES operator commands.

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

Note: **`Show JCL`** is not an operator command like the other Job Monitor commands (**`Hold`**, **`Release`**, **`Cancel`**, and **`Purge`**), so the limitations below do not apply to **`Show JCL`**.

Job Monitor issues all JES operator commands that you or another user requests through an extended MCS (EMCS) console, whose name is controlled with the `CONSOLE_NAME` directive, as documented in "Job Monitor configuration file BLZJCNFG" on page 72.

With this setup, you or the security administrator can define granular command execution permits using the `OPERCMDS` and `CONSOLE` classes.

- To use an EMCS console, you must have, at minimum, READ authority to the `MVS.MCSOPER.console-name` profile in the `OPERCMDS` class.

  Note: If you do not define a profile, the system will grant the authority request.
- To execute a JES operator command, you must have sufficient authority to access the `JES%.**` profile in the `OPERCMDS` class.

  Note: If you do not define a profile, or if the `OPERCMDS` class is not active, JES will fail the command.
- You can also require that a user must use Job Monitor to perform the operator command by specifying `WHEN(CONSOLE(JMON))` on the **PERMIT** definition. The `CONSOLE` class must be active for this setup to work.

  Note: It is sufficient for the `CONSOLE` to be active. No profiles are checked for EMCS consoles.

Your security software prevents the assumption of the identity of the Job Monitor server by creating a JMON console from a TSO session. Even though the console can be created, the point of entry is different: Job Monitor versus TSO. JES commands that you issue from this console will fail the security check if your security is set up as documented in this information center, and if you or another user does not have authority to JES commands through other means.

Note: If the console name is already in use, Job Monitor cannot create the console when a command must be executed. To prevent this, you can set the `GEN_CONSOLE_NAME=ON` directive in the Job Monitor configuration file, or you can define security profiles to stop TSO users from creating a console.

The following sample RACF commands prevent all unauthorized users 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: Users who are not authorized to make these operator commands can still submit jobs and read job output through Job Monitor if they have sufficient authority to access profiles that might protect these resources, like those in the `JESINPUT`, `JESJOBS`, and `JESSPOOL` classes.

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

Access to spool files:

You can access and manage permissions for all spool files through Job Monitor.

By default, you have browse access to all spool files through Job Monitor. You can change this with the `LIMIT_VIEW` directive, as documented in "Job Monitor configuration file BLZJCNFG" on page 72.

Table 15. Job Monitor 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 Job Monitor configuration file `BLZJCNFG`. If they need access to a wider range of jobs, but not to all jobs, use the standard spool file protection features of your security product, like the `JESSPOOL` class.

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

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

Job Monitor configuration file:

Modify your `BLZJCNFG` Job Monitor configuration file to customize security directives and command limits.

The directives in the `BLZJCNFG` Job Monitor configuration file impact security setup. The security administrator and systems programmer should decide what the settings should be for the following directives:

- `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 81.
- `LIMIT_VIEW={USERID | NOLIMIT}`

  Define which spool files can be browsed. For more information, see "Access to spool files" on page 84.

Note: Details on the `BLZJCNFG` directives are available in "Job Monitor configuration file BLZJCNFG" on page 72.

Security definitions:

Review information about the Job Monitor started task and defining JES command security.

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

- "Define the Job Monitor started task"
- "Define JES command security" on page 86

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

Define the Job Monitor started task:

Use various RACF commands to create `BLZJMON` started tasks, and to protect started task user IDs.

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

-
  ```
  ADDGROUP STCGROUP OMVS(GID(#group-id))
  DATA('GROUP WITH OMVS SEGMENT FOR STARTED TASKS')
  ```

-
  ```
  ADDUSER STCJMON DFLTGROUP(STCGROUP) NOPASSWORD NAME('JOB MONITOR')
  OMVS(UID(#user-id-jmon) HOME(/tmp) PROGRAM(/bin/sh) NOASSIZEMAX)
  DATA('RATIONAL TEAM CONCERT')
  ```

-
  ```
  RDEFINE STARTED BLZJJCL.* DATA('JOB MONITOR')
  STDATA(USER(STCJMON) GROUP(STCGROUP) TRUSTED(NO))
  ```

-
  ```
  SETROPTS RACLIST(STARTED) REFRESH
  ```

Note: Ensure that the started task user ID is protected by specifying the `NOPASSWORD` keyword.

Define JES command security:

Use various RACF commands to limit user access to JES commands in Job Monitor.

Job Monitor issues all JES operator commands through an extended MCS (EMCS) console, whose name is controlled with the `CONSOLE_NAME` directive, as documented in "Job Monitor configuration file BLZJCNFG" on page 72.

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

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

Notes:

1. Usage of the console is permitted if no `MVS.MCSOPER.#console` profile is defined.
2. 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.
3. 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: If you define JES commands with universal access `NONE` in your security software, you might impact other applications and operations. Test this before you activate it on a production system.

Table 16 and Table 17 show the operator commands issued for JES2 and JES3, and the discrete security profiles that you can use to protect them.

Table 16. 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 17. JES3 Job Monitor operator commands

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

Table 17. JES3 Job Monitor operator commands (continued)

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

Notes:

1. The **`Hold`**, **`Release`**, **`Cancel`**, and **`Purge`** JES operator commands, and the **`Show JCL`** command, can be performed only against spool files that the user ID owns, unless `LIMIT_COMMANDS=` with value `LIMITED` or `NOLIMIT` is specified in the Job Monitor configuration file. Refer to "Actions against jobs: target limitations" on page 81 for more information.
2. You can browse any spool file, unless `LIMIT_VIEW=USERID` is defined in the Job Monitor configuration file. Refer to "Access to spool files" on page 84 for more information.
3. User who are not authorized for these operator commands can still submit jobs and read job output through Job Monitor, provided that they have sufficient authority to profiles that might protect these resources, like those in the `JESINPUT`, `JESJOBS` and `JESSPOOL` classes.

Your security software prevents the assumption of the identity of the Job Monitor server by creating a JMON console from a TSO session. Even though the console can be created, the point of entry is different: 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 information center, and if you do not have authority to the JES commands through other means.

Additional setup options that depend on the Build System Toolkit on z/OS

In addition to the ISPF client and Enterprise Extensions builds, promotions and deployments, these components depend on the Build System Toolkit running on z/OS.

About this task

If you have installed the Build System Toolkit on z/OS and you plan to set up the following components, review the following procedures:

Procedure

1. If you are setting up the Jazz Build Engine on z/OS, see Running the Jazz Build Engine on z/OS.
2. If you are installing the Rational Developer for System z integration feature, see Installing and configuring the integration.
3. If you are setting up the context-aware search engine on z/OS, see Installing the context-aware search engine on z/OS at http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0/topic/com.ibm.team.scm.doc/topics/t_install_z_search_engine.html
4. If you will be running JCL-based builds and need a new Job Monitor installed, see Using the Rational Build Agent and Job Monitor to run builds using JCL at

http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0/topic/com.ibm.team.build.doc/topics/rtczRBAandJM.html

Using the Jazz Build Engine on z/OS

This section describes how to submit a build for native z/OS artifacts and obtain results using the Jazz Build Engine.

Refer to the following topics for information about using the Jazz Build Engine on z/OS:

Running the Jazz Build Engine on z/OS

Follow these steps to process your build request on z/OS using the Jazz Build Engine.

Before you begin

The Jazz Build Engine is installed when you install the Build System Toolkit on z/OS. Setting up the build engine is optional. Some build templates for z/OS require the Rational Build Agent, *not* the Jazz Build Engine. You need the Jazz Build Engine only for builds that you define using a Jazz Build Engine template, such as the **Command Line - Jazz Build Engine** template. For more information, see Build templates available for z/OS.

Before performing the setup steps, review the Building with Jazz Team Build lesson in the *Get started with Rational Team Concert* tutorial.

About this task

Complete the following steps before starting the build engine:

Procedure

1. Use the Eclipse client to create a build engine definition on the Rational Team Concert server. This "registers" the build engine with the server, which needs to know such build engine characteristics as the build engine ID, supported build definitions, and so forth. These characteristics are saved in the build engine definition.
2. Two user IDs (which might not be the same) are required to run the Jazz Build Engine on z/OS: First, the build engine JCL specifies a user ID and password that are used to connect and authenticate to the server to perform build and SCM operations. Second, the user ID the *BLZBENG* JCL requires on z/OS determines what z/OS data set and file system authority should be used on the z/OS build system.
   a. Confirm that the user ID specified in the *BLZBENG* JCL is a valid user for your server, and that it has the required credentials to extract the artifacts to build.
   b. Confirm that the user ID assigned to the *BLZBENG* job has the authority to create the necessary artifacts on z/OS.
3. **Optional:** Create an encrypted password file for the build engine to use to prevent casual observation. For information on creating an encrypted password file, see Creating a password file using the sample BLZBPASS job.
4. Edit the BLZBENG member and modify the settings using the instructions in the JCL.
5. To start the build engine, submit the BLZBENG JCL. The job must remain active and the following messages must end in the STDOUT: `<Time and date>`

`Running build loop...` and `<`*`Time and date`*`> Waiting for request.` If these messages are not shown in the STDOUT, check the SYSOUT to identify the issues that occurred when you initialized the build engine.

6. To end the build engine, stop or purge the job.

Creating a password file using the sample BLZBPASS job

You can use a sample JCL *BLZBPASS* job, which is provided with the SMP/E package, to protect any file with an encrypted password.

About this task

You can create a file containing an encrypted password using a sample JCL BLZBPASS job provided with the SMP/E package. You can specify the file location and the encrypted password using the parameters specified in the job. You can use this password file in your build scripts. View the sample JCL for more details.

Note: The encrypted password file format changed in Rational Team Concert version 4.0, but you do **not** have to create your password files again, because the code can read password files from previous releases. The new format is binary-compatible among platforms. You can create a new password file for z/OS using the sample BLZBPASS job, or by using a Jazz Build Engine on another platform (such as Windows), and then transferring the output to z/OS in binary mode.

Procedure

1. Configure member BLZBPASS in *hlq*.SBLZSAMP using the instructions contained in the sample JCL.
2. Submit the modified JCL. The job must end with a return code 0.
3. Do not save the modified JCL with any password associated with it.

Installing the context-aware search tool

The context-aware search tool helps you find information in your code that is based on free text searches. When you configure the context-aware search tool, you must install, define, and start a context-aware search engine, and then configure your search parameters. The context-aware search engine manages the search index, as defined by your search parameters.

To configure the context-aware search engine, you must install and configure the following components:

- Rational Team Concert Eclipse client
- Jazz Team Server.

Note: To configure and run a search engine, you must have a Developer for IBM Enterprise Platforms client access license (CAL).

The context-aware search engine requires an IBM Java run time environment (JRE) compatible with Java 1.6. Use the JDK included with the Rational Team Concert Eclipse client for your platform, if available. The JDK is in the `jdk` subdirectory if you used the IBM Installation Manager to install it, and in the `eclipse/jdk` subdirectory if you installed the client from a compressed file.

Note:

1. If you are running Rational Team Concert Version 4.x with clustered servers, existing Rational Team Concert Version 3.x search engines cannot be used.
2. The information about an IBM JRE compatible with Java 1.6 being required does not apply to IBM i.

The minimum amount of memory recommended for running the context-aware search engine is 512 MB. Performance improves if you can allocate more memory to its process. To change the memory settings, change the **MAX_MEM** or **-Xmx** parameter in the script file for your platform. The default value is `1300M`, which allocates a maximum of 1300 MB for the Java heap size. To use more memory, a 64-bit JRE might be necessary.

The amount of required disk-space depends on the number of indexed components and on the number of files in those components. The initial installation requires 140 MB for most platforms.

Installing the context-aware search engine on z/OS systems

Before you can use the context-aware search on z/OS systems, you must install and start the search engine on your operating system. Install the context-aware search engine on z/OS by configuring members of *hlq*.SBLZSAMP and submitting the JCL.

About this task

Complete the following steps before you start the search engine:

Procedure

1. Configure member BLZBPASS in *hlq*.SBLZSAMP, where *hlq* is the high-level qualifier you used during the SMP/E installation, using the instructions contained in the sample JCL.
2. Submit the modified JCL. You can also use this password file in your build scripts. Do not save the modified JCL with any password associated with it. The job must end with a return code of 0.
3. Configure member BLZWASE in *hlq*.SBLZSAMP using the instructions contained in the sample JCL.
4. Submit the modified JCL. The job must continue to run and the following messages must display in the STDOUT output:

```
[2010-10-26 15:18:24] [----------------]: Service loop started.
[2010-10-26 15:18:24] [----------------]: Not logged in to repository -
                        https://uri...
[2010-10-26 15:18:28] [----------------]: user is logged in to repository -
                        https://uri...
[2010-10-26 15:18:36] [------Begin-----]: Initialization
[2010-10-26 15:18:41] [-------End------]: Initialization
[2010-10-26 15:18:44] [----------------]: Configuration handler is active
[2010-10-26 15:18:44] [------Begin-----]: Synchronizing configurations
                               with server
[2010-10-26 15:18:46] [----------------]: Index handler is active
[2010-10-26 15:18:46] [----------------]: Synchronizing indices with server
[2010-10-26 15:18:47] [-------End------]: Synchronizing configurations
                               with server
```

5. If these messages are not displayed in the STDOUT output, other messages in the STDOUT output should indicate any problems that occurred during the search engine initialization. Additional information is available under

dir/`sjxLog/exceptions`, where *dir* is the path you specified for the **`-baseOutputDir`** parameter in the modified JCL.

6. To stop the search engine, cancel or delete the job.

Installing and configuring the Rational Developer for System z integration feature

There are additional installation and configuration steps required to integrate Rational Developer for System z and Rational Team Concert.

Installing the file agent RSE miner

Perform the steps described in this topic to install and configure the file agent Remote Systems Explorer (RSE) miner.

Before you begin

Follow the installation and configuration steps for the RSE server and daemon in the *Rational Developer for System z Host Configuration Guide, (SC23-7658)*, which can be found at the Rational Developer for System z Library page.

These steps require that you have already installed the Build System Toolkit for System z.

Procedure

1. Locate the `rsed.envvars` file in the `RSE configuration` directory. See *rsed.envvars, RSE* in the *Rational Developer for System z Host Configuration Guide* for instructions, which can be found at the Rational Developer for System z Library page.
2. Save a copy of this file using the following command: **`cp rsed.envvars rsed.envvars.org`**
3. Edit the `rsed.envvars` file using the **`ISPF 3.17`**, the **`oedit`** command in OMVS (**`oedit rsed.envvars`**), or the Rational Developer for System z editor if you use Remote Systems Explorer.
4. You must designate a class path statement, library path statement, and a working directory location. Find the *CLASSPATH* assignment, `CLASSPATH=.:$CLASSPATH`, in the `rsed.envvars` file. Immediately ahead of the *CLASSPATH* assignment, copy the sample member *BLZENVAR* from hlq.*SBLZSAMP*. *hlq* is the high-level qualifier specified during the SMP/E installation. The member contains the Rational Team Concert *CLASSPATH* assignment. The sample member also contains stubbed declarations of both the *LIB_PATH* and *SCM_WORK* environment variables. For clarification on what to set these environment variables to, review the comments in `rsed.envvars` file.

   Note:

   - Apply the class path statement addition for Rational Team Concert to every instance of the RSE server for which you want to run Rational Team Concert. After performing the configuration steps, you must re-create any RSE connections that existed previously in your Rational Developer for System z clients.
   - Developers using the Rational Developer for System z and Rational Team Concert integration feature require write access to the directory pointed to by *SCM_WORK* (and at least read and execute access to its parent directories).

The following is an example of the Java `rsed.envvars` file. It includes modification prompts to show you how and where to customize your own `rsed.envvars` file, according to your build environment.

```
#
# Insert the following Classpath allocation in the rsed.envvars
# just prior to the CLASSPATH=.:$CLASSPATH statement, then
# uncomment and update it to reference the install path for RTC
#
# Rational Team Concert
#
#CLASSPATH=$CLASSPATH:@pathPrefix@/usr/lpp/jazz/v4.0.6/buildsystem
                          /buildtoolkit/*

# Uncomment and update the following statement to reference the install
# path for RTC
#LIBPATH=$LIBPATH:@pathPrefix@/usr/lpp/jazz/v4.0.6/buildsystem
                          /buildtoolkit

# Uncomment and update the following statement to reference the working
# directory.
# Working directories are typically stored in /u/jazz406. You can change
# this directory.
# Make sure that the userId under which the Rational Developer for
# System z Remote Explorer daemon runs has write access permission to
# this directory
#SCM_WORK=@workPath@/rdz
```

5. Stop and restart the RSE server and daemon.

Installing the integrated client

Installing the Rational Team Concert Integration feature requires that you install it after Rational Developer for System z is installed, or with Rational Developer for System z (if Rational Team Concert is already installed.)

Before you begin

These instructions assume that you are installing the Rational Team Concert client before you install the Rational Developer for System z client. If you already have Rational Developer for System z client installed and you want to add the Rational Team Concert client, see "Installing Rational Team Concert on Rational Developer for System z" on page 93.

Notes:

- Rational Team Concert version 4.0.6 requires Rational Developer for System z version 9.0. For additional system requirements information, see Hardware and software requirements.
- Not all Eclipse-based products work with all versions of the Eclipse integrated development environment (IDE). If you are installing Rational Team Concert or Rational Developer for System z into an existing Eclipse IDE, check the system requirements for both products to make sure that they are compatible.

Procedure

1. Install the Rational Team Concert client using the Rational Team Concert launchpad.
2. Install the Rational Developer for System z client using the Rational Developer for System z launchpad.

   Note: Make sure that you select the same Installation Manager package that you used in your Rational Team Concert client installation.
3. Select **Rational Team Concert Integration feature** in the Installation Manager.

   Important: This integration feature is only available for installation if Rational Team Concert is already installed into the same Installation Manager package into which you are installing Rational Developer for System z.
4. Follow the remaining Installation Manager wizard instructions to complete the installation.

Installing Rational Team Concert on Rational Developer for System z

If you already have the Rational Developer for System z client installed, you can install the Rational Team Concert client into your existing installation by modifying the existing client installation package using IBM Installation Manager.

Before you begin

If you have not installed the Rational Team Concert client, you cannot install the Rational Team Concert Integration feature. Perform the following steps to install Rational Team Concert and the extension that is the Rational Team Concert integration feature.

Procedure

1. Install the Rational Team Concert client using the Rational Team Concert launchpad.
2. Select the option to install into the same package group as the Rational Developer for System z client.

   Note: Not all Eclipse-based products work with all versions of the Eclipse integrated development environment (IDE). If you are installing Rational Team Concert client or Rational Developer for System z client into an existing Eclipse IDE, check the system requirements for both products to make sure that they are compatible.
3. Open IBM Installation Manager.
4. From the list of available installation packages, select IBM Rational Team Concert Integration feature.
5. Follow the remaining Installation Manager wizard instructions to finish the installation.

Setting up additional debug parameters on z/OS

This information helps you debug problems with the integration of Rational Developer for System z and Rational Team Concert on z/OS.

If you have problems with using the Rational Developer for System z and Rational Team Concert integration, IBM support might ask you to enable some additional debug settings. Instructions for setting up the debug settings follow:

1. Locate the directory which contains the rsed.envvars configuration file used by the Rational Developer for System z Remote System Explorer (RSE) daemon.
2. Create a file named log4j.properties in the same directory and give it the same Unix System Services (USS) file ownership and permissions as the rsed.envvars file.
3. Add the following statements or other statements as directed by IBM support to the log4j.properties file:

   ```
   log4j.rootLogger=DEBUG, stdout
   log4j.appender.stdout=org.apache.log4j.RollingFileAppender
   log4j.appender.stdout.file.MaxFileSize=10MB
   log4j.appender.stdout.file.MaxBackupIndex=5
   log4j.appender.stdout.file=/tmp/RDzRTCtrace.txt
   log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
   log4j.appender.stdout.layout.ConversionPattern=%-4r %-5p Ý%t¨ %c %3x - %
   log4j.rootCategory=ERROR, A1
   # Set to the minimum for our package (DEBUG)
   log4j.logger.com.ibm.teamz.fileagent=DEBUG,
   log4j.logger.com.ibm.teamz.fileagent.miner=DEBUG,
   log4j.logger.com.ibm.teamz.fileagent.miner.common=DEBUG,
   log4j.logger.com.ibm.teamz.fileagent.internal.jazzscm.JazzSCMResource=DEBUG
   ```
4. Restart the RSE daemon. A trace file is written to the location specified on the `log4j.appender.stdout.file` statement in the log4j.properties file.

   Note: To disable the trace, you can delete or rename the log4j.properties file and restart the RSE daemon.

Installing the Rational Team Concert ISPF client

The ISPF client is installed as part of the SMP/E installation. The ISPF client enables you to edit, check-in, deliver, and build code stored in a Rational Team Concert repository.

The Rational Team Concert Interactive System Productivity Facility (ISPF) client provides an ISPF interface to perform various Rational Team Concert functions. You can use the interface to edit, check-in, deliver, and build code that is stored in a Rational Team Concert repository.

The ISPF client is installed as part of the Build System Toolkit during the SMP/E installation of FMID HRBT406. Ensure the SMP/E installation of this FMID has been completed.

For a video demonstration of the ISPF client, see Using the ISPF client with z/OS UNIX System Services on jazz.net at http://jazz.net/library/video/772.

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

ISPF client security

The ISPF client must be secure so that only authorized users can access stored files.

The Rational Team Concert ISPF client provides mainframe access to source code that is stored in an Rational Team Concert repository that might or might not be on the z/OS machine to which the user is already authenticated. Therefore, the

ISPF daemon needs to validate connection requests and provide secure communication between the host and the repository.

The security mechanism used by the Rational Team Concert ISPF daemon relies on the file system where it resides to be secure. This implies that only trusted system administrators should be able to update the program libraries and configuration files.

To create ISPF client security definitions, customize and submit sample member BLZRACFT, which has sample RACF and z/OS UNIX commands, to create the basic security definitions for Rational Team Concert. BLZRACFT is located in *hlq*.SBLZSAMP, unless you have copied it to another library for customization.The user submitting this job must have security administrator privileges, such as being RACF SPECIAL.

Note: The sample BLZRACFT job holds more than just RACF commands. The last step of the security definitions consists of making various z/OS UNIX files program controlled. Depending on the policies at your site, this might be a task for the system programmer and not the security administrator.

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

The user ID under which the ISPF daemon runs (as defined in BLZRACFT) should be added to the SAF Group that has write access to the CCM working directories when sample configuration JOB BLZCPBTK was submitted.

Through the ISPF client, you provide the user's Rational Team Concert userid and password to the ISPF daemon through the dialogs. The authentication data provided by the client is only used once, during initial connection login. Once a user ID is authenticated, the user ID and self-generated PassTickets are used for all actions that require authentication. When you log out or exit the ISPF client, the authentication connection is lost and you must authenticate again the next time you use the ISPF client.

Sample RACF commands for these steps are provided in sample job BLZRACFT, but they are discussed in more detail in the topics included in the following list:

- "Security settings and classes for the ISPF client"
- "OMVS segment for ISPF client users" on page 96
- "Data set profiles for the ISPF client" on page 96
- "ISPF daemon started task on System z" on page 98
- "ISPF daemon as a secure z/OS UNIX server" on page 98
- "Application protection for the ISPF daemon" on page 98
- "ISPF PassTicket support" on page 99
- "z/OS UNIX program-controlled files for the ISPF daemon" on page 101

Note: Refer to the *RACF Command Language Reference (SA22–7687)* for more information on RACF commands.

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

Security settings and classes for the ISPF client

Activate security settings and classes for the ISPF client with RACF commands.

Rational Team Concert uses several security mechanisms to ensure a secure and controlled host 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 application protection for Rational Team Concert ISPF daemon
  ```
  SETROPTS GENERIC(APPL)
  SETROPTS CLASSACT(APPL) RACLIST(APPL)
  ```
- Activate secured signon using PassTickets for the ISPF daemon
  ```
  SETROPTS GENERIC(PTKTDATA)
  SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
  ```

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

OMVS segment for ISPF client users

Submit commands to define a RACF OMVS segment or equivalent for each ISPF client user.

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

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

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

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

Although it is not recommended, you can use the shared OMVS segment defined in the BPX.DEFAULT.USER profile of the FACILITY class to fulfill the OMVS segment requirement for Rational Team Concert.

In addition, the user ID assigned to the started task for the ISPF daemon must also have a non-zero z/OS UNIX user ID (UID), home directory, and shell command defined.

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

Data set profiles for the ISPF client

Provide access to Rational Team Concert data sets for users.

READ access for users and ALTER for system programmers is sufficient for most Rational Team Concert data sets. Ask the system programmer who installed and configured the product for the correct data set names. BLZ is the default high-level qualifier.

Replace the #sysprog placeholder with valid user IDs or RACF group names in the following sample RACF commands:.

- `ADDGROUP (BLZ) OWNER(IBMUSER) SUPGROUP(SYS1)`
  `DATA('RATIONAL TEAM CONCERT - HLQ STUB')`
- `ADDSD 'BLZ.**' UACC(READ)`
  `DATA('RATIONAL TEAM CONCERT')`
- `PERMIT 'BLZ.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)`
- `SETROPTS GENERIC(DATASET) REFRESH`

Note: The sample commands here assume that enhanced generic naming (EGN) is active. EGN allows the ** qualifier to represent any number of qualifiers in the DATASET class. Substitute ** with * if EGN is not active on your system. Refer to *Security Server RACF Security Administrator's Guide*, (SA22-7683), for more information on EGN.

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

- uacc(none) data set protection
  - `ADDGROUP (BLZ)`
    `DATA('RATIONAL TEAM CONCERT - HLQ STUB')`
    `OWNER(IBMUSER) SUPGROUP(SYS1)`
  - `ADDSD BLZ.**' UACC(NONE)`
    `DATA('RATIONAL TEAM CONCERT')`
  - `ADDSD 'BLZ.SBLZLOAD' UACC(NONE)`
    `DATA('RATIONAL TEAM CONCERT')`
  - `ADDSD 'BLZ.SBLZEXEC' UACC(NONE)`
    `DATA('RATIONAL TEAM CONCERT')`
  - `ADDSD 'BLZ.SBLZMENU' UACC(NONE)`
    `DATA('RATIONAL TEAM CONCERT')`
  - `ADDSD 'BLZ.SBLZMENU' UACC(NONE)`
    `DATA('RATIONAL TEAM CONCERT')`
  - `ADDSD 'BLZ.SBLZSAMP' UACC(NONE)`
    `DATA('RATIONAL TEAM CONCERT')`
- Permit system programmer to manage all libraries
  - `'BLZ.** CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)`
  - `PERMIT 'BLZ.SBLZLOAD' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)`
  - `PERMIT 'BLZ.SBLZEXEC' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)`
  - `PERMIT 'BLZ.SBLZMENU' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)`
  - `PERMIT 'BLZ.SBLZPENU' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)`
  - `PERMIT 'BLZ.SBLZSAMP' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)`
- Permit clients to access the load and exec libraries
  - `PERMIT 'BLZ.SBLZLOAD' CLASS(DATASET) ACCESS(READ) ID(*)`
  - `PERMIT 'BLZ.SBLZEXEC' CLASS(DATASET) ACCESS(READ) ID(*)`
  - `PERMIT 'BLZ.SBLZMENU' CLASS(DATASET) ACCESS(READ) ID(*)`
  - `PERMIT 'BLZ.SBLZPENU' CLASS(DATASET) ACCESS(READ) ID(*)`
  - `PERMIT 'BLZ.SBLZSAMP' CLASS(DATASET) ACCESS(READ) ID(*)`
- Activate security profiles
  - `SETROPTS GENERIC(DATASET) REFRESH`

When controlling READ access to system data sets, you must provide Rational Team Concert users permission to READ the REXX.V1R4M0.SEAGLPA data set.

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

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

ISPF daemon started task on System z

Submit RACF commands to create the BLZISPFD and BLZISPFS started tasks for the ISPF daemon.

The following sample RACF commands create the BLZISPFD and BLZISPFS started tasks, with protected user ID (STCISPF) and group STCGROUP assigned to them. Replace the #group-id and #user-id-* placeholders with valid OMVS IDs.

- `ADDGROUP STCGROUP OMVS(GID(#group-id))`
  `DATA('GROUP WITH OMVS SEGMENT FOR STARTED TASKS')`
- `ADDUSER STCISPF DFLTGRP(STCGROUP) NOPASSWORD NAME('RTC - ISPF DAEMON')`
  `OMVS(UID(#user-id-ispf) HOME(/tmp) PROGRAM(/bin/sh) ASSIZEMAX(2147483647))`
  `DATA('RATIONAL TEAM CONCERT')`
- `RDEFINE STARTED BLZISPFD.* DATA('RTC - ISPF DAEMON - START')`
  `STDATA(USER(STCISPF) GROUP(STCGROUP) TRUSTED(NO))`
- `RDEFINE STARTED BLZISPFS.* DATA('RTC - ISPF DAEMON - STOP')`
  `STDATA(USER(STCISPF) GROUP(STCGROUP) TRUSTED(NO))`
- `SETROPTS RACLIST(STARTED) REFRESH`

Notes:

1. Ensure that the started tasks user IDs are protected by specifying the NOPASSWORD keyword.
2. Ensure that the ISPF daemon has a unique OMVS user ID due to the z/OS UNIX related privileges granted to this user ID.
3. The ISPF daemon requires a large address space size (2GB) for proper operation. It is advised to set this value in the ASSIZEMAX variable of the OMVS segment for user ID STCISPF. This to ensure that the ISPF daemon gets the required region size, regardless of changes to MAXASSIZE in SYS1.PARMLIB(BPXPRMxx).

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

ISPF daemon as a secure z/OS UNIX server

Provide the ISPF daemon UPDATE access to the BPX.SERVER profile to manage the security environment.

The ISPF daemon requires UPDATE access to the BPX.SERVER profile to create or delete the security environment for the client's thread.

Use the following commands to permit access by the ISPF daemon:

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

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

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

Application protection for the ISPF daemon

Enable user verification for client connections.

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

Use the following commands to enable user verification by the ISPF daemon:

-
  ```
  RDEFINE APPL BLZAPPL UACC(READ)
  DATA('RATIONAL TEAM CONCERT')
  ```

-
  ```
  SETROPTS RACLIST(APPL) REFRESH
  ```

Notes:

1. The client connection request fails if the profile is not defined, or when the user has no READ access to the profile.
2. As described in more detail in “ISPF PassTicket support,” the ISPF daemon supports the usage of an application ID other than BLZAPPL. The APPL class definition must match the PTKTDATA class definition as well as the actual application ID used by the ISPF daemon.

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

ISPF PassTicket support

PassTickets establish thread security within the ISPF daemon.

Client passwords are only used to verify identities during connections. Afterwards, PassTickets are used to maintain thread security. PassTickets are system generated passwords with a lifespan of about 10 minutes. The generated PassTickets are based on a secret key. This key is a 64-bit number (16 hex characters). Replace the key16 placeholder with a user-supplied 16 character hex string (characters 0-9 and A-F) in the sample RACF commands below.

-
  ```
  RDEFINE PTKTDATA BLZAPPL UACC(NONE) SSIGNON(KEYMASKED(key16))
  APPLDATA('NO REPLAY PROTECTION – DO NOT CHANGE')
  DATA('RATIONAL TEAM CONCERT')
  ```

  The following example shows the command with the key16 value replaced:

  ```
  RDEFINE PTKTDATA BLZAPPL UACC(NONE) -
  DATA('RATIONAL TEAM CONCERT') -
  APPLDATA('NO REPLAY PROTECTION - DO NOT CHANGE') -
  SSIGNON(KEYMASKED(0123456789ABCDEF))
  ```

-
  ```
  RDEFINE PTKTDATA IRRPTAUTH.BLZAPPL.* UACC(NONE)
  DATA('RATIONAL TEAM CONCERT')
  ```

-
  ```
  PERMIT IRRPTAUTH.BLZAPPL.* CLASS(PTKTDATA) ACCESS(UPDATE) ID(STCISPF)
  ```

-
  ```
  SETROPTS RACLIST(PTKTDATA) REFRESH
  ```

Notes:

1. 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.
2. Substitute the wildcard (*) in the IRRPTAUTH.BLZAPPL.* definition with a valid user ID mask to limit the user IDs for which the ISPF daemon can generate a PassTicket.
3. Depending on your RACF settings, the user defining a profile might also be on the access list of the profile. You should remove this permission for the PTKTDATA profiles.
4. If the system has a cryptographic product installed and available, you can encrypt the secured signon application key for added protection.

Use the KEYENCRYPTED keyword instead of KEYMASKED. Refer to *Security Server RACF Security Administrator's Guide*, (SA22-7683), for more information.

5. If you want to use an application ID other than BLZAPPL, change BLZAPPL to an application ID that meets your needs. Ensure that you change the definition in the APPL class shown in "Application protection for the ISPF daemon" on page 98. Also, ensure that you set the _ISPF_DAEMON_APPLID property in the ispfdmn.conf file to the changed value before you start the ISPF daemon. See "ISPF daemon configuration file (ispfdmn.conf)" on page 103 for details.

After logon, PassTickets are used to establish thread security within the ISPF daemon. 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.

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

1. The ISPF client connects to ISPF daemon port 4152.
2. The ISPF daemon authenticates the client, using the credentials presented by the client.
3. The ISPF daemon creates a unique client ID and an ISPF server thread.
4. The ISPF daemon generates a PassTicket and creates a security environment for the client, using the PassTicket as the password.
5. The ISPF daemon validates the client using the client ID.
6. The ISPF daemon uses a newly generated PassTicket as the 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. The ISPF daemon generates and uses a PassTicket each time a password is required, resulting in a (temporary) valid password for the client.

Using PassTickets allows the ISPF daemon to set up a user-specific security environment, without the need of storing all user IDs and passwords in a table, which could be compromised. Using PassTickets 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 use PassTickets. These profiles are application specific and do not impact your current system setup.

PassTickets being application specific implies that the ISPF daemon must use a unique application ID (APPLID). By default, the ISPF daemon uses BLZAPPL as the APPLID.

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

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

z/OS UNIX program-controlled files for the ISPF daemon

The ISPF daemon needs UPDATE access to the BPX.SERVER profile to manage the security environment.

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

The ISPF daemon server uses RACF's Java shared library (`/usr/lib/libIRRRacf.so`) as well as a number of Rational Team Concert programs.

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

Notes:

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

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

ISPF daemon configuration

The SMP/E installation of the ISPF client daemon provides a configuration file, and shell scripts and sample proclib members to help you configure, start, and stop the ISPF daemon.

The ISPF client daemon components are created as part of the SMP/E installation, and includes the following components:

ispfdmn.sh
Shell script to start the daemon

ispfstop.sh
Shell script to stop the daemon

ispfdmn.conf
Configuration file containing customizable settings for the ISPF daemon

BLZISPFD
Sample proclib member in *hlq*.SBLZSAMP to start the daemon as a started task (sample JCL)

BLZISPFS
Sample proclib member in *hlq*.SBLZSAMP to stop the daemon as a started task (sample JCL)

BLZRACFT
: Sample JCL member in *hlq*.SBLZSAMP to create required security definitions for the daemon (sample JCL)

Unless you specified a different location when you customized and submitted job *hlq*.SBLZSAMP(BLZCPBTK), the ispfdmn.sh, ispfstop.sh, and ispfdmn.conf files were copied to a configuration directory. By default, the directory is /etc/jazz406/ccm.

Note: The daemon on z/OS needs to be restarted after a server rename. This is true for the ISPF daemon on Version 3 or Version 4. The ISPF daemon on Version 4 should disconnect itself after the server rename, but the ISPF daemon on Version 3 might not.

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

Started tasks for the ISPF daemon

Customize the sample started task members SBLZSAMP(BLZISPFD) and SBLZSAMP(BLZISPFS) for starting and stopping the ISPF daemon.

The ISPF daemon has two started task procedures: one to start the daemon and one to stop the daemon in the system procedure library defined by your JES subsystem. In the instructions below, the IBM default procedure library, SYS1.PROCLIB, is used.

Customize the sample started task members *hlq*.SBLZSAMP(BLZISPFD) and *hlq*.SBLZSAMP(BLZISPFS), as described within the members, and copy them to SYS1.PROCLIB. These instructions assume PROCLIB member names of BLZISPFD to start the daemon and BLZISPFS to stop the daemon have been created; however, you can use any names. Ensure that the PROCLIB members match the started task definitions in the BLZRACFT job in *hlq*.SBLZSAMP that you have already submitted.

As shown in the code sample below, you must provide:

1. The ISPF daemon port. The default port is 4152.
2. The home directory where Rational Team Concert is installed. The default directory is `/usr/lpp/jazz/v4.0.6`.
3. The location of the configuration files. The default location is `/etc/jazz406/ccm`.
4. The location of the directory for the temporary files. The default location is `/tmp`.

```
//*
//* ISPF DAEMON - Start
//*
//BLZISPFD PROC PORT=4152,
//            HOME='/usr/lpp/jazz/v4.0.6',
//            CNFG='/etc/jazz406/ccm',
//            WORK='/tmp'
//BLZISPFD EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//            PARM='PGM &HOME./ispfclient/bin/ispfdmn.sh &PORT &CNFG &WORK'
//STDOUT  DD   SYSOUT=*
//STDERR  DD   SYSOUT=*
//       PEND
//*
```

Notes:

1. Refer to "ISPF daemon START (S) command" on page 105 for additional information.

2. To ensure that the ISPF daemon shuts down cleanly at system shutdown time, the BLZISPFS started task should be used. You should add this started task to your system shutdown procedures.

The maximum length for the PARM variable is 100 characters, which might cause problems if you use custom directory names. To bypass this problem, you can either:

- Use symbolic links

  Symbolic links can be used as shorthand for a long directory name. The following sample z/OS UNIX command defines a symbolic link (`/usr/lpp/jazz`) to another directory (`/long/directory/name/usr/lpp/jazz`).

  ```
  ln -s /long/directory/name/usr/lpp/jazz /usr/lpp/jazz
  ```
- Use STDIN

  When the PARM field is empty, BPXBATSL starts a z/OS UNIX shell and runs the shell script provided by STDIN. STDIN must be a z/OS UNIX file (allocated as ORDONLY). Using STDIN disables the use of variables for the PROC parameters. The shell executes the shell logon scripts `/etc/profile` and `$HOME/.profile`.

  To use this method, you must update the startup JCL to contain code similar to the following sample:

  ```
  //*
  //* ISPF DAEMON - USING STDIN
  //*
  //BLZISPFD PROC CNFG='/etc/jazz406'
  //*
  //RSE EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT
  //STDOUT DD SYSOUT=*
  //STDERR DD SYSOUT=*
  //STDIN DD PATHOPTS=(ORDONLY),PATH='&CNFG./ispfdmn.stdin.sh'
  // PEND
  //*
  ```

  You must also create the shell script (`/etc/jazz/ispfdmn.stdin.sh` in this example) to start the ISPF daemon. The content of this script should contain code similar to the following sample:

  ```
  /long/directory/name/usr/lpp/jazz/v4.0.6/ispfclient/bin/ispfdmn.sh 4197
                                /etc/jazz/ccm
  ```

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

ISPF daemon configuration file (ispfdmn.conf)

Customize the ISPF daemon by specifying settings in the ispfdmn.conf file.

The ISPF daemon uses the definitions in ispfdmn.conf. This file is located in `/etc/jazz406/ccm`, unless you specified a different location when you customized and submitted job *hlq*.SBLZSAMP(BLZCPBTK) as part of the installation and configuration of the Build System Toolkit. If you have not previously run BLZCPBTK, run it now. For more information, see “Installing and configuring the Build System Toolkit on z/OS systems” on page 45.

You can edit the ispfdmn.conf file with ISPF option 3.17. The ispfdmn.conf file must be customized to match your system environment. Comment lines start with a pound sign (#), when using a US code page. Data lines can only have a directive and its assigned value, and comments are not allowed on the same line. Line continuations are not supported, and spaces around the equal sign (=) are not supported.

Note: The BLZISPFD started task must be restarted to pick up any changes you make.

The following definitions are required:

_ISPF_CLEANUP_INTERVAL
: This environment variable is used in conjunction with the _ISPF_CLEANUP_THRESHHOLD environment variable. The cleanup interval logs out any sessions that have been inactive for the specified time within the specified threshold time. For example if the _ISPF_CLEANUP_INTERVAL is set to 900000, which is 15 minutes, and the _ISPF_CLEANUP_THRESHHOLD is set to 86400000, which equals 24 hours, the daemon scans every 15 minutes and discards any sessions that have not been active in the last 24 hours.

  Set this value to -1 to disable this cleanup.

_ISPF_CLEANUP_THRESHHOLD
: See the _ISPF_CLEANUP_INTERVAL definition for an explanation of how this environment variable is used. The default is 86400000, which equals 24 hours.

  Set this value to -1 to disable this cleanup.

_ISPF_DAEMON_APPLID=BLZAPPL
: The application ID that the ISPF daemon uses for passticket security support. The application ID specified here must match what you define in the APPL class definition and the PTKTDATA class definition as shown in job BLZRACFT in *hlq*.SBLZSAMP.

_ISPF_DAEMONCONFIG_DIR
: Specifies the absolute path to the ISPF daemon configuration directory, for example `/u/`*`username`*`/work/configuration/daemonconfig`.

_ISPF_REGISTRY_DIR
: Specifies the path to the daemon registry directory. This is a hierarchical file system (HFS) location where the daemon writes its location information. For example, you could set this to `/u/work/ccm`.

JAVA_HOME
: Specifies the path to the Java home directory. The default is `/usr/lpp/java/J6.0_64`. Change to match your Java installation.

PATH_TO_IRRRACF
: Specifies the path to the location of the Java interface to your security product, IRRRacf.jar. The default is `/usr/include/java_classes/`. Change to match your security software setup.

  Note: Since z/OS 1.10, `/usr/include/java_classes/IRRRacf.jar` is part of SAF, which ships with base z/OS, so it is available also to non-RACF customers.

RTC_HOME
: Specifies the path to the Rational Team Concert home directory. The default is `/usr/lpp/jazz/v4.0.6/`. Change to match your Rational Team Concert installation.

SCM_WORK
: The path to the directory where the metadata are stored for PDSEs. The metadata are stored under ${SCM_WORK}/SCM. Set this to your work directory, which is `/u/jazz406/ccm` by default. Developers using the

Rational Team Concert ISPF client require write access to the directory pointed to by SCM_WORK (and at least read and execute access to its parent directories).

The following definitions are optional:

_CEE_DMPTARG
: Specifies the absolute path to the directory where CEEDUMPs are written, for example `/u/`*`username`*`/work/ccm`.

_ISPF_CLEAN
: When you start the daemon, if you want to force Eclipse to reinitialize the cached data, set this definition to **–clean**.

_ISPF_CLIENT_LOG_MAX
: Specifies the initial maximum number of client trace log files for a user. The default number of trace log files is 3.

  To disable client trace logging, set this value to zero.

_ISPF_CLIENT_LOG_OVER
: Specifies whether individual users can override the default values for **_ISPF_CLIENT_LOG_MAX** and **_ISPF_CLIENT_LOG_SIZE** set in the ISPF daemon configuration file. Valid values are `YES` and `NO`. The default setting is `YES`. Setting this value to `NO` disables individual users from changing the logging settings in the ISPF client preferences.

_ISPF_CLIENT_LOG_SIZE
: Specifies the initial maximum client trace log file size for a user, in bytes. The default is 100000 bytes.

  When the first log file reaches the size specified or allowed to default, subsequent log entries are logged into a new log file.

_ISPF_DAEMON_LOG_CONF
: Specifies the absolute path to where the ISPF daemon trace log files are stored, for example `/u/`*`username`*`/conf/ccm/commons-logging.properties`.

The log files are stored in a path that is relative to the value of **_ISPF_REGISTRY_DIR**, for example *daemon registry directory*`/logs/`*userid*. To view the log files, open Help (F1) and select **`1. Display logs`**.

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

ISPF daemon START (S) command

The START command provides necessary settings for starting the ISPF daemon.

Use the START command to dynamically start a started task (STC). The abbreviated version of the command is the letter S.

```
►►──┬─START─┬──procname──┬────────────────────────┬──────────────────────►
    └─S─────┘            │             ┌─4152─┐   │
                         └─,──PORT──=──┴─port─┴───┘

►──┬─────────────────────────────────────────┬───────────────────────────►
   │            ┌─'usr/lpp/jazz/v4.0.6'─┐    │
   └─,──HOME──=─┴─'install_path'────────┴────┘
```

```
►─┬──────────────────────────────────────┬──┬──────────────────────────────────┬──────►◄
  │                ┌'etc/jazz406/ccm'─┐  │  │                ┌'/tmp'─────────┐  │
  └,──CNFG──=──────┴'config_path'─────┴──┘  └,──WORK──=──────┴'tempdir_path'──┴──┘
```

procname
The name of the member in a procedure library used to start or stop the server. The default name used during the host configuration to start the daemon is BLZISPFD. The default name used during the host configuration to stop the daemon is BLZISPFS.

When the BLZISPFS started task is started, it runs the process to bring down the BLZISPFD started task, cleaning up any daemon connections. After connection clean up, both the BLZISPFD and BLZISPFS started tasks complete.

Note: It is important to use the BLZISPFS started task to bring the daemon down, as it ensures proper cleanup.

PORT The port used by the ISPF daemon for the clients to connect. The default is 4152.

HOME
Path prefix and the mandatory /usr/lpp/jazz/v4.0.6 used to install Rational Team Concert. The default is '/usr/lpp/jazz/v4.0.6'.

Note: The z/OS UNIX path is case sensitive and it must be enclosed in single quotes (') to preserve lower case characters.

CNFG Absolute location of the configuration files stored in z/OS UNIX. The default is '/etc/jazz406/ccm'.

Note: The z/OS UNIX path is case sensitive and it must be enclosed in single quotes (') to preserve lower case characters.

WORK
The name of the directory where temporary files are stored. The default is '/tmp'.

Note: The z/OS UNIX path is case sensitive and it must be enclosed in single quotes (') to preserve lower case characters.

The syntax diagram shows you how to specify a command so that the operating system can correctly interpret what you type. Read the syntax diagram from left to right and from top to bottom, following the horizontal line (the main path).

The following symbols are used in syntax diagrams:

Symbol	**Description**
>>	Marks the beginning of the syntax diagram.
>	Indicates that the syntax diagram is continued.
|	Marks the beginning and end of a fragment or part of the syntax diagram.
><	The end of the syntax diagram.

The following types of operands are used in syntax diagrams:
- Required operands are displayed on the main path line:

```
►►──REQUIRED_OPERAND──────────────────────────────────────────────────►◄
```

- Optional operands are displayed below the main path line:

```
►►──┬──────────────────┬──────────────────────────────────────────────►◄
    └─OPTIONAL_OPERAND─┘
```

- Default operands are displayed above the main path line:

```
    ┌─DEFAULT_OPERAND─┐
►►──┴─────────────────┴───────────────────────────────────────────────►◄
```

Operands are classified as keywords or variables:

- Keywords are constants that must be provided. If the keyword appears in the syntax diagram in both uppercase and lowercase, the uppercase portion is the abbreviation for the keyword (for example, KEYword). Keywords are not case-sensitive. You can code them in uppercase or lowercase.
- Variables are italicized, appear in lowercase letters, and represent names or values you supply. For example, a data set name is a variable. Variables can be case sensitive.

In the following example, the USER command is a keyword. The required variable parameter is user_id, and the optional variable parameter is password. Replace the variable parameters with your own values:

```
►►──USER──user_id──┬──────────┬───────────────────────────────────────►◄
                   └─password─┘
```

If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, equal signs, and blank spaces), you must code the character as part of the syntax. In this example, you must code OPERAND=(001 0.001):

```
►►──OPERAND──=──(──001 0.001──)───────────────────────────────────────►◄
```

An arrow returning to the left in a group of operands means that more than one can be selected, or that a single one can be repeated:

```
►►──┬──────────────────────┬──────────────────────────────────────────►◄
    ├─REPEATABLE_OPERAND_1─┤
    ├─REPEATABLE_OPERAND_2─┤
    └─<────────────────────┘
```

If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead:

```
►►──┤ The first line of a syntax diagram is longer than one line ├──────────►

►─┤ The continuation of the subcommands, parameters, or both ├──────────►◄
```

Some diagrams might contain syntax fragments, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram:

```
►►──┤ Syntax fragment ├──────────────────────────────────────────────►◄
```

Syntax fragment:

```
├──1ST_OPERAND──,──2ND_OPERAND──,──3RD_OPERAND──────────────────────────┤
```

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

Configuring and running the ISPF client on z/OS systems

ISPF client libraries are created as part of the SMP/E installation. The libraries contain files to help configure and run the ISPF client.

Complete these tasks to customize the ISPF client. The security settings in "ISPF client security" on page 94 are required to use the ISPF client and daemon.

The Rational Team Concert ISPF client libraries are created as part of the SMP/E installation. When you run the REXX procedure for starting the ISPF client, the libraries are dynamically added to your TSO/ISPF session. The libraries are assigned only when needed by using LIBDEF and ALTLIB services. This method ensures that existing TSO/ISPF logon procedures do not need to be changed.

The components of the ISPF client dialog are delivered in these libraries:

SBLZEXEC
REXX EXECs

SBLZLOAD
Load modules

SBLZM*xxx*
ISPF messages

SBLZP*xxx*
ISPF panels

*hlq***.SBLZSAMP**
Sample JCL members

where *xxx* identifies the national language. For example, SBLZPENU is the ISPF panel library for US English.

Note: Latest entries in the Jazz.net library on installing Rational Team Concert

Starting the ISPF client

There are several methods for starting the ISPF clients.

Start the ISPF client dialog using the **BLZ** REXX executable code. You can run the executable code in several ways:

- From the TSO command processor panel:

  On the TSO command processor panel, enter `EX '<hlq>.SBLZEXEC(BLZ)''<hlq><language>'`

- Added to an ISPF menu:

  Set &ZSEL to `'CMD(EX "<hlq>.SBLZEXEC(BLZ)" "<hlq><language>") NOCHECK'`. NOCHECK supports the entry of concatenated commands through the direct option (trail). On the calling panel, also specify &ZTRAIL=.TRAIL.
- Added as a command in the SYSPROC concatenation:

  Create an EXEC in the SYSPROC concatenation (for example, BLZISPF) that starts the BLZ EXEC with the parameters hardcoded:

  ```
  /* Start the RTC ISPF Client */
  ex 'BLZ.SBLZEXEC(BLZ)' 'BLZ'
  ```

  After creating the executable code, run the code from the command line. Enter the following command: `TSO %BLZISPF`.

  If the command is added to a command table, enter `%BLZISPF`.

Note: BLZ.SBLZEXEC(BLZ) contains two variables that you might want to customize for your environment. Edit the executable code and change the variable values if your installation differs from the stated defaults:
- BLZICONV identifies the location of the iconv installation and the default is `/bin/iconv`.
- BLZLPBAK is the hostname for the daemon host and the default is 127.0.0.1.

The ISPF client command `BLZ` accepts two parameters:

`HLQ`
: The data set name high level qualifiers for the ISPF client data sets.

`Language`
: (Optional.) Identifies the national language. Currently, the ISPF client only supports the following languages:

ENU	U.S. English, which is the default if **Language** is omitted
ENP	U.S. English Uppercase
JPN	Japanese

Chapter 4. What to do next

Additional setup tips and tasks

After your installation and configuration is complete on z/OS, complete the following tasks:

About this task

Review the resources listed below to complete your setup.

Procedure

1. Review Tips for installing and setting up CLM products on z/OS at https://jazz.net/wiki/bin/view/Main/EESetup on jazz.net.
2. For information about components installed on other systems, review Deployment and installation considerations at http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0m6/topic/com.ibm.jazz.install.doc/topics/c_deployment_considerations.html in the Rational solution for Collaborative Lifecycle Management information center.
3. To install a client, see Installing a Rational Team Concert client at http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0m6/topic/com.ibm.jazz.install.doc/topics/c_client_installation.html in the Rational solution for Collaborative Lifecycle Management information center.

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created

Trademarks

Appendix B. z/OS sample members

The following tables list the member names, types, and usage of the samples that are included in the MVS data set hlq.SBLZSAMP.

Table 18. Samples for initial set up

Member name	Type	Usage
BLZCPBFA	JCL	Creates and customizes z/OS UNIX System Services (USS) files and directories for the Rational Build Agent
BLZCPBTK	JCL	Creates and customizes USS files and directories for the
BLZCPCCM	JCL	Creates and customizes USS files and directories for the Change and Configuration Management (CCM) application
BLZCPJTS	JCL	Creates and customizes USS files and directories for Jazz(tm) Team Server
BLZCPOPT	JCL	Creates and customizes USS files and directories for the Apache Tomcat application server
BLZCPQM	JCL	Creates and customizes USS files and directories for the Quality Management (QM) application
BLZCPRM	JCL	Creates and customizes USS files and directories for the Requirements Management (RM) application

Table 19. Resource Access Control Facility (RACF) samples

Member name	Type	Usage
BLZRACF	JCL	General RACF sample definitions
BLZRACFT	JCL	RACF sample definitions

Table 20. Repository tools samples

Member name	Type	Usage
BLZADDTB	JCL	Adds or updates Jazz Team Server, CCM, or QM DB2 database tables
BLZADDT2	JCL	Adds or updates Jazz Team Server, CCM, or QM Derby database tables
BLZCCMU	JCL	Updates CCM configuration files during an upgrade
BLZCREDB	JCL	Creates DB2 database tables or data warehouse tables for Jazz Team Server and the CCM and QM applications
BLZCRED2	JCL	Creates Derby database tables or data warehouse tables for Jazz Team Server and the CCM and QM applications
BLZEXPOR	JCL	Exports the DB2 repository for Jazz Team Server and the CCM and QM applications
BLZEXPO2	JCL	Exports the Derby repository for Jazz Team Server and the CCM and QM applications
BLZIMPOR	JCL	Imports the database repository to DB2 for Jazz Team Server and the CCM and QM applications.
BLZIMPO2	JCL	Imports the database repository to Derby for Jazz Team Server and the CCM and QM applications.
BLZREIDX	JCL	Performs query triple store and Lucene text store reindexing

Table 20. Repository tools samples (continued)

Member name	Type	Usage
BLZRTIDX	JCL	Performs text reindexing
BLZJTSU	JCL	Upgrades Jazz Team Server configuration files
BLZDWHUP	JCL	Upgrades DB2 data warehouse tables
BLZDWHU2	JCL	Upgrades Derby data warehouse tables
BLZGENMP	JCL	Generates a stub mapping file for use with server renames
BLZRENAM	JCL	Imports a list of URL prefix mappings for use during the rename of a server that is backed by a DB2 database
BLZQMU	JCL	Upgrades the QM application
BLZRMU	JCL	Upgrades the RM application
BLZRPOTL	JCL	Can be modified to run any repository tool command

Table 21. Samples for starting and stopping components

Member name	Type	Usage
BLZBENG	JCL	Starts the
BLZBFA	JCL	Starts the Rational Build Agent as a started task
BLZGWSRV	JCL	Starts the Enterprise REST Gateway server
BLZISPFD	JCL	Starts the ISPF daemon server
BLZISPFS	JCL	Stops the ISPF daemon server
BLZJJCL	JCL	Starts the Job Monitor
BLZSRVC	JCL	Starts Tomcat with Derby repositories
BLZSRVD	JCL	Starts Tomcat with DB2 repositories
BLZWASE	JCL	Starts the context-aware search engine

Table 22. Samples for Rational Developer for System z and Job Monitor

Member name	Type	Usage
BLZJCNFG	Configuration file	Sample Job Monitor configuration file to use with the Rational Developer for System z integration feature
BLZENVAR	Configuration file	Sample configuration file to use with the Rational Developer for System z integration feature and Java 1.6
BLZTSO	Job Monitor sample	Used with Job Monitor

Table 23. Enterprise Extensions promotion and deployment sample executable files

Member name	Type	Usage
BLZBKPZP	REXX	Deployment backup sample EXEC.
BLZDEPZP	REXX	Deployment sample EXEC.
BLZPKGZP	REXX	Deployment package sample EXEC.

Table 24. Miscellaneous samples

Member name	Type	Usage
BLZBPASS	JCL	Creates an encrypted password file to use with the , Rational Build Agent, or Rational Developer for System z integration feature.
BLZCSAMP	REXX	Compiles a simple COBOL program
BLZDTLEX	REXX	Processes allocations and calls the ISPF DTL conversion utility (ISPDTLC)
BLZGPASS	JCL	Generates a password file for the Enterprise REST Gateway
BLZGTWY	JCL	Starts the ISPF Gateway
BLZGWCLI	REXX	Enterprise REST Gateway client
BLZGWTST	REXX	Enterprise REST Gateway test stub
BLZSCLM1	REXX	A sample SCLM user exit to use with

Table 25. DB2 repository backup

Member name	Type	Usage
BLZBIND	JCL	DB2 BIND for the plans for BLZUNLD and BLZREPR programs
BLZDB2UN	JCL	UNLOAD the Rational solution for CLM in six steps: 1. Runs the BLZUNLD program to generate UNLOAD statements. 2. Prints the non-LOB table UNLOAD statements for reference. 3. Prints the LOB table UNLOAD statements for reference. 4. Creates the temporary HFS directories for the LOB UNLOAD. 5. Runs the non-LOB table UNLOAD. 6. Runs the LOB table UNLOAD.
BLZSYSIN	REXX	Merges the SYSIN files and then edits the merged SYSIN file: • To change the schema prefix, if required • To change RESUME YES to RESUME NO REPLACE
BLZDB2LD	JCL	UNLOAD the Rational solution for CLM in five steps: 1. Deletes the merged SYSIN file. 2. Runs the BLZSYSIN program. 3. Runs the LOAD. 4. Runs BLZREPR program to create REPAIR statements. 5. Runs the REPAIR.

Printed in USA