IBM Engineering Lifecycle Management 7.0.3

Installation guide for z/OS

Note

Before using this information, be sure to read the general information under “Notices for IBM Engineering Lifecycle Management” on page 123.

This edition applies to version 7.0.3 of the IBM® Engineering Lifecycle Management products and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Installing on z/OS

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 IBM Engineering Lifecycle Management (ELM) 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 ELM applications on z/OS, such as IBM Engineering Workflow Management (EWM), IBM Engineering Test Management (ETM), and IBM Engineering Requirements Management DOORS® Next (DOORS Next). 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 IBM Engineering Lifecycle Management v7.0.3 program directories for z/OS at https://www.ibm.com/support/pages/node/6479339.

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 Engineering Workflow Management 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/v7.0.3, where *@pathPrefix@* is the path prefix that you specified during the SMP/E installation.

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

HRCC703 - 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.

HRBA703 - Rational® Build Agent
Install this FMID if you plan to run Enterprise Extensions, JCL, or command line builds on z/OS.

HRBT703 - Engineering Workflow Management - Build System Toolkit
Install this FMID if you plan to run any of the following functions on z/OS:

- Enterprise builds
- Jazz Build Engine
- ISPF client
- Mass import tool
- Promotion
- Packaging and deployment
- Integration with IBM Developer for z/OS

HRCM703 - Engineering Workflow Management - Change and Configuration Management (CCM)
Install this FMID if you plan to use the server CCM capabilities of Engineering Workflow Management on z/OS.

HRDV703 - Engineering Workflow Management - IBM Developer for z/OS subset
Install this FMID in these cases:

- If you do not have IBM Developer for z/OS installed and you plan to submit or monitor JCL-based builds on z/OS.
- If you do not have IBM Developer for z/OS installed and you plan to use the Engineering Workflow Management compilation results view as part of your enterprise builds for z/OS compilations.

HRGC703 - Global Configuration Management
Install this FMID if you plan to use the functions of Global Configuration Management on z/OS.

HRJS703 - Jazz Team Server
Install this FMID if you plan to run Jazz Team Server on z/OS. HRJS703 includes the server executable files that are installed to the z/OS UNIX System Services. HRJS703 also includes the startup JCL installed to a z/OS data set.

HRLI703 - Link Index Provider
Install this FMID if you plan to use the functions of Link Index Provider on z/OS.

HRQM703 - Engineering Test Management - Quality Management (QM)
Install this FMID if you plan to use the server QM capabilities of Engineering Test Management on z/OS.

HRRE703 - IBM Engineering Lifecycle Optimization - Engineering Insights (ENI)
Install this FMID if you plan to use the functions of Engineering Insights on z/OS.

HRRM703 - Requirements Management (RM)
Install this FMID if you plan to use the server RM capabilities of Engineering Requirements Management DOORS Next.

HRRS703 - IBM Jazz Reporting Service
Install this FMID if you plan to use the functions of the IBM Jazz Reporting service, including the Data Collection Component (DCC), Lifecycle Query Engine (LQE), or Report Builder.

HRWL703 - IBM WebSphere Liberty
Install this FMID if you plan to run the server by using the IBM WebSphere Liberty on z/OS. This FMID also includes WebSphere Application Server webserver plug-ins to be used with IBM HTTP Server on z/OS. Refer to documentation of WebSphere Application Server for information on configuring the HTTP server and plug-ins.

Installation process on z/OS

The process to install on z/OS is as follows.

1. Complete the SMP/E installation.
2. Finish the customization on z/OS.
3. If you installed Jazz Team Server on z/OS, configure Jazz Team Server.

4. Configure any ELM applications that you installed on z/OS.
5. Install the Engineering Workflow Management client on your computer.

Checklist for installing the Engineering Lifecycle Management on z/OS systems

Before you begin

Review the system requirements on Jazz.net at https://jazz.net/deployment-wiki-home.jsp.

About this task

Installations of the Engineering Lifecycle Management on z/OS can be a hybrid of applications installed on z/OS and other systems. Use this checklist as a guide to complete the required steps for the applications you plan to install.

Procedure

Complete these general installation tasks:

1. Use SMP/E to install the required components. For details, see "SMP/E installation process" on page 1.
2. Review the additional considerations for system programmers. For details, see "Installation information for z/OS system programmers" on page 4.
3. Set up the JZOS batch launcher. For details, see "Setting up the JZOS batch launcher" on page 9.
4. Review the security considerations for ELM products on z/OS. For details, see "Security on z/OS systems" on page 9.

Complete these installation tasks, which are specific to installing Jazz Team Server and the ELM applications on z/OS:

For instructions, see *Chapter 2: Setting up the server on z/OS*.

5. Tailor and run BLZCP* sample jobs to create and customize additional directories.

   If you are running IBM WebSphere Liberty on z/OS, tailor and run BLZCPWLP. For more information about the IBM WebSphere Liberty on z/OS and BLZCPWLP customization, see "Configuring Jazz Team Server and the applications for the Engineering Lifecycle Management to use IBM WebSphere Liberty on z/OS" on page 57.
6. If you are creating a DB2® database on z/OS, set up Db2.
   a) Create high-level Db2 objects.
   b) Tailor the `teamserver.properties` file for Db2for z/OS.
   c) Run BLZCREDB to create Db2 for z/OS tables.
7. Run Jazz Team Server and the ELM applications on IBM WebSphere Liberty on z/OS.

If you are installing the Build System Toolkit on z/OS, including the ISPF client and the SCM command line, complete these tasks:

For instructions, see *Chapter 3. Installing optional tools, clients, and samples on z/OS*.

8. Tailor and run BLZCPBTK.
9. If you are running Enterprise Extensions builds, promotions, or deployments, configure the Legacy ISPF gateway. Optionally, you can configure Enterprise Extensions builds to support the Interactive ISPF gateway. For more information, see "Configuring the Interactive ISPF gateway for Enterprise Extensions build support" on page 69.

   The Enterprise Extensions builds, promotions, and deployments depend on the installation and configuration of the Rational Build Agent, as described later in this section.

After you install the Build System Toolkit, you can set up an ISPF client, the Jazz Build Engine, the IBM Developer for z/OS integration feature and the Rational Build Agent. For instructions, see the following sections.

10. If you plan to use the ISPF client, complete these steps:

    For instructions, see *Chapter 3. Installing optional tools, clients, and samples on z/OS*.

    a) Set up ISPF client security.

    b) Define the ISPF client daemon-started tasks.

    c) Configure the ISPF client daemon.

    d) Start the ISPF client daemon.

    e) Start the ISPF client.

11. If you plan to use Jazz Build Engine on z/OS, set up the engine.
12. If you plan to run builds, promotions, or deployments on z/OS with the Rational Build Agent, complete these tasks:

    a) Tailor and run BLZCPBFA.

    b) If you are using a port other than 5555, modify the bfagent.conf file.

    c) If you are using Ant with Enterprise Extensions, complete these tasks:

        i) Modify the startbfa.sh file.

        ii) Create a password file by using the sample BLZBPASS job.

    d) Start the Rational Build Agent as a started task, inetd task, or shell command.

13. After you install the Rational Build Agent, if you plan to run JCL-based builds and need to install a new Job Monitor, set up the Rational Build Agent and Job Monitor.

Installation information for z/OS system programmers

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

Before you install the IBM Engineering Lifecycle Management (ELM) products on a z/OS system, review the following information:

1. Hardware and software requirements in the ELM Documentation
2. “SMP/E installation process” on page 1
3. “Security on z/OS systems” on page 9
4. “PARMLIB changes” on page 4
5. “TCP/IP ports” on page 6
6. “z/OS sample members” on page 6

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 IBM Engineering Workflow Management (EWM) 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

The library hlq.SBLZAUTH, which contains the BLZPASTK module, must be made available through the LNKLST definition.

Note: The steps of the LNKLST definitions are needed only if you plan to use the EWM ISPF client.

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 primary 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:

- Modules `BLZJMON` and `BLZENF70` in the *hlq*`.SBLZAUTH` load library, where *hlq* is the high-level qualifier you used during SMP/E installation. These modules must be APF-authorized only if you do not have IBM Developer for z/OS installed and if you intend to run z/OS builds that submit JCL directly.
- 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 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 Engineering Workflow Management SMP/E package, this port is the default port for the Job Monitor.

IBM WebSphere Liberty server ports
Typically, you configure the WebSphere Liberty ports when you configure the application server.

z/OS sample members

These tables list the member names, types, and usage of the samples that are included in the following MVS data set: *hlq*.SBLZSAMP.

Table 1. Samples for initial setup

Member name	Type	Usage
BLZCPBFA	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Rational Build Agent
BLZCPBTK	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Build System Toolkit
BLZCPCCM	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Change and Configuration Management (CCM) application
BLZCPDCC	JCL	Creates and customizes z/OS UNIX System Services directories for the Data Collection Component
BLZCPGC	JCL	Creates and customizes z/OS UNIX System Services directories for Global Configuration Management
BLZCPJTS	JCL	Creates and customizes z/OS UNIX System Services files and directories for Jazz Team Server
BLZCPLDX	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Link Index Provider
BLZCPLQE	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Lifecycle Query Engine
BLZCPPVM	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Build Manager
BLZCPQM	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Quality Management (QM) application
BLZCPRE	JCL	Creates and customizes z/OS UNIX System Services files and directories for the IBM Engineering Lifecycle Optimization - Engineering Insights (ENI) application
BLZCPRM	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Requirements Management (RM) application

Table 1. Samples for initial setup (continued)		
Member name	**Type**	**Usage**
BLZCPRS	JCL	Creates and customizes z/OS UNIX System Services files and directories for the Report Builder
BLZCPWLP	JCL	Creates and customizes z/OS UNIX System Services files and directories for the IBM WebSphere Liberty server

Table 2. Resource Access Control Facility (RACF) samples		
Member name	**Type**	**Usage**
BLZRACFL	JCL	General RACF® sample definitions and IBM WebSphere Liberty server definitions
BLZRACFT	JCL	Build System Toolkit RACF sample definitions

Table 3. Repository tools samples		
Member name	**Type**	**Usage**
BLZADDTB	JCL	Adds or updates Jazz Team Server, ELM applications, or Derby database tables
BLZCCMU	JCL	Updates CCM configuration files during an upgrade
BLZCREDB	JCL	Creates Db2 or Derby database tables or data warehouse tables for Jazz Team Server and the ELM applications
BLZDCCU	JCL	Upgrades Data Collection Component configuration files
BLZDWHLD	JCL	Loads data warehouse tables
BLZDWHUN	JCL	Unloads data warehouse tables
BLZEXPOR	JCL	Exports the Db2 repository for Jazz Team Server and the ELM applications
BLZGCU	JCL	Upgrades Global Configuration Management configuration files
BLZIMPOR	JCL	Imports the database repository to Db2 for Jazz Team Server and the ELM applications
BLZJTSU	JCL	Upgrades Jazz Team Server configuration files
BLZDWHUP	JCL	Upgrades Db2 or Derby data warehouse tables
BLZQMU	JCL	Upgrades the QM application
BLZRELMU	JCL	Upgrades Engineering Insights configuration files
BLZRMU	JCL	Upgrades the RM application
BLZRPOTL	JCL	Can be modified to run any repository tool command

Table 4. Samples for starting and stopping components		
Member name	**Type**	**Usage**
BLZBENG	JCL	Starts the Jazz Build Engine
BLZBFA	JCL	Starts the Rational Build Agent as a started task
BLZISPFD	JCL	Starts the Engineering Workflow Management ISPF daemon server
BLZISPFS	JCL	Stops the Engineering Workflow Management ISPF daemon server

Table 4. Samples for starting and stopping components (continued)		
Member name	**Type**	**Usage**
BLZJJCL	JCL	Starts the Job Monitor
BLZPVM	JCL	Starts the Engineering Workflow Management Build Manager daemon

Table 5. Samples for IBM Developer for z/OS and Job Monitor		
Member name	**Type**	**Usage**
BLZJCNFG	Configuration file	Sample Job Monitor configuration file to use with the IBM Developer for z/OS integration feature
BLZZEXPL	Configuration file	Sample configuration file to use with the IBM Developer for z/OS integration feature
BLZTSO	Job Monitor sample	Used with Job Monitor

Table 6. 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 7. Miscellaneous samples		
Member name	**Type**	**Usage**
BLZBPASS	JCL	Creates an encrypted password file to use with the Jazz Build Engine, Rational Build Agent, or IBM Developer for z/OS integration feature.
BLZDTLRX	REXX	Processes allocations and calls the ISPF DTL conversion utility (ISPDTLC)
BLZGTWY	JCL	Starts the Legacy ISPF Gateway
BLZIVP	JCL	Engineering Workflow Management installation verification process
BLZRSCMP	JCL	Sample JCL for compacting the RS database
BLZZANGL	JCL	Start the IBM WebSphere Liberty angel process
BLZZSRV	JCL	Start the IBM WebSphere Liberty server

Table 8. Db2 repository backup		
Member name	**Type**	**Usage**
BLZBIND	JCL	Db2 BIND for the plans for BLZUNLD and BLZREPR programs
BLZDB2UN	JCL	UNLOAD ELM 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. Runs the non-LOB table UNLOAD. 5. Runs the LOB table UNLOAD.

Table 8. Db2 repository backup (continued)		
Member name	**Type**	**Usage**
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 ELM 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.

Setting up the JZOS batch launcher

Many of the sample JCL members used to configure and run the IBM Engineering Lifecycle Management (ELM) applications on z/OS require the IBM JZOS Java batch launcher.

About this task

The IBM JZOS Batch Toolkit for z/OS SDKs includes a batch launcher and toolkit for Java applications that run on z/OS. The JZOS batch launcher is an MVS batch program that configures and launches the Java virtual machine (JVM). Configure the JZOS batch launcher to support running sample JCL for the Engineering Lifecycle Management.

Procedure

Configuring the JZOS batch launcher requires the following basic tasks. For details, see JZOS Batch Launcher and Toolkit (https://www.ibm.com/docs/en/semeru-runtime-ce-z/11?topic=guide-jzos-batch-launcher-toolkit).

1. Add the batch launcher load module into a z/OS PDSE (partitioned data set extended).
2. Customize the sample start PROC and add it to the appropriate PROCLIB.

Security on z/OS systems

When you install Jazz Team Server and the other IBM Engineering Lifecycle Management (ELM) applications on z/OS systems, several tasks are required to make the ELM functions secure and available.

These instructions are intended for people who are installing a combination of the Jazz Team Server, any of the ELM applications, the IBM Engineering Workflow Management (EWM) 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 ELM applications, the EWM 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 ELM components on z/OS.

z/OS UNIX System Services directory protection
To install and configure the ELM components on z/OS, you use three main directories and associated subdirectories that need appropriate user and group-level permissions:

1. Product binary files: Installed by SMP/E, typically to a directory such as `/usr/lpp/jazz/v7.0.3`
2. Configuration directories: Created by running sample configuration jobs to create and populate a directory such as `/etc/jazz703`
3. Working directories: Created by running sample configuration jobs to create and populate a directory such as `/u/jazz703`

Database access
If you are running Jazz Team Server and ELM applications on z/OS, you must provide access from the server to Db2 z/OS databases for the applications and data warehouse.

Started tasks
Started tasks and associated user IDs can be defined for use with the IBM WebSphere Liberty Server profile, the daemon used to support the ISPF client, and Rational Build Agent.

Additional server profile requirements and EJBROLEs
Additional RACF requirements are defined if you are running the server on z/OS as well as RACF EJBROLEs to control user access

Three sample members are provided from the *hlq*.SBLZSAMP library, where *hlq* is the high-level qualifier that was specified during the SMP/E installation:

- BLZRACFL: This sample member is intended for when you are also planning to use the IBM WebSphere Liberty on z/OS for the application server and is installed with HRWL703.
- BLZRACFT: This sample member is for the Build System Toolkit and Rational Build Agent, and is installed with SMP/E FMID HRBT703.

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" on page 10: 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 12: Read this topic if you are deploying Jazz Team Server and the ELM applications on z/OS.
- "Security for the Build System Toolkit and Rational Build Agent on z/OS systems" on page 13: 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 IBM Engineering Lifecycle Management (ELM) 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 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 z/OS UNIX Systems Services directories. Other users have READ and EXECUTE access. You can set up additional security protections.

Creation of z/OS UNIX System Services 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. Directories are created with the sticky bit turned on.

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 ELM target and distribution libraries so that you can protect the HLQ by using RACF. Users who work with z/OS functions such as the IBM Engineering Workflow Management (EWM) 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 ELM applications, you can see an example of RACF statements in the instructions that are provided with the BLZRACFL job in *hlq*.SBLZSAMP, where *hlq* is the high-level qualifier that was specified during the SMP/E installation. Review these jobs carefully.

If you are installing the EWM Build System Toolkit, the same RACF commands are provided in the BLZRACFT job in *hlq*.SBLZSAMP.

For most EWM 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('EWM - HLQ STUB')

#  general data set protection
  LISTDSD PREFIX(BLZ) ALL
  ADDSD 'BLZ.**' -
   UACC(READ) DATA('EWM')
  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.

BLZRACFL 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/jazz703` and `/u/jazz703`. 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 BLZRACFL 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 z/OS UNIX System Services configuration and work directories by limiting access to the directories that contain them. For example, access to `/etc/jazz703` can be restricted if the user or group does not have READ access to `/etc`.

Jazz Team Server security on z/OS systems

If you are installing Jazz Team Server on z/OS systems, you must consider several security options. These options help secure your data and provide appropriate access to different types of users.

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 "RACF security on z/OS systems" on page 10, 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 connect to the data warehouse.

IBM WebSphere Liberty security setup

If you plan to run Jazz Team Server with a IBM WebSphere Liberty, you must set up several RACF profiles. Specifically, the user ID under which the application server runs must have READ and WRITE access to the IBM Engineering Lifecycle Management (ELM) server configuration and work directories. Therefore, the user ID must be added to the JAZZCONF and JAZZWORK GROUPs.

In addition, each ELM 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 IBM WebSphere Liberty. 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 Engineering Lifecycle Management applications on a z/OS system with IBM WebSphere Liberty on z/OS" on page 55
- "Setting up user security on a z/OS system by using RACF" on page 55

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

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

If you are using the Build System Toolkit and Rational Build Agent with the applications for the IBM Engineering Lifecycle Management (ELM) on z/OS, you must consider several security options. These options help secure your data and provide appropriate access to different types of users.

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 10, 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 IBM Engineering Workflow Management (EWM) 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.

EWM 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 Engineering Workflow Management 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 (GC19-3623).

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).

Rational Build Agent security setup

You can run the Rational Build Agent in several ways. For details, see Security for the Rational Build Agent in the EWM Documentation. The 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')
 #  UID(0) is not required

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

 RDEFINE STARTED BLZBFA.* DATA('EWM - 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 IBM Developer for z/OS 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) NOASSIZEMAX)
ALTGROUP #group-name OMVS(GID(#group-identifier))
```

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

Engineering Workflow Management 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.

For additional details, see Using the Rational Build Agent and Job Monitor to run builds using JCL in the Engineering Lifecycle Management Documentation.

IBM Developer for z/OS integration feature security setup

If you plan to run the IBM Developer for z/OS integration feature with Engineering Workflow Management, 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)
```

Setting permission bits for temporary files and logs in z/OS UNIX System Services

When you run dependency builds, promotions, or packaging, load files from SCM to a z/OS UNIX System Services directory during a dependency build, perform an SCM operation during an ISPF client session, use the IBM Developer for z/OS Integration Load and Share function, or perform an SCM operation with the z/OS Unix Shell view from RSE, Engineering Workflow Management generates z/OS configured files, logs, and temporary files. If you want to change the permission bits for these types of files, take the following actions:

- Login to (or have the system administrator login to) `ccm/admin` with a browser. Navigate to **Server** > **Configurations** > **Advanced Properties**. Under **Build Agent** verify that the **Default File Permission** is set to the desired value. The default value is 775.
- Log in to the z/OS UNIX Shell, and change or add a *umask* value in `/etc/profile`.
- Login to z/OS UNIX Shell, and verify the **umask** specified in `install_dir/ispfclient/bin/ispfdmn.sh` is set to the desired value. The default umask value is `0002`, as shown in the following code sample:

```
 ------------------------------------------------------------
cd $RTC_HOME/scmtools/eclipse
umask 0002
mask=`umask`
echo
------------------------------------------------------------
echo starting ISPF daemon on port $_ISPF_DAEMON_PORT ... -- $(date)

                                      umask=$mask echo
------------------------------------------------------------
```

  Restart the ISPF daemon.
- Set **umask** in `rsed.envvars`.

Installing the configuration utility

The IBM Engineering Lifecycle Management (ELM) configuration utility is based on ISPF to aid in the configuration of the components of ELM on the z/OS platform.

About this task

Before configuring your Jazz Team Server and the IBM Engineering Lifecycle Management (ELM) products on z/OS, you must ensure that the SMP/E installation has ended successfully. Because of the many optional configuration components in ELM, knowing how to install everything can be confusing. The configuration utility enables you to select the components you want to install and creates a work flow based list of items to run that perform the configuration. The work flow is presented in a logical order of items to run. You can add additional configuration items or even create completely new configurations.

The installation of the ELM configuration utility is performed by the SMP/E installation of FMID HRCC703, which is the Common Components FMID. For more information, see "SMP/E installation process" on page 1. The SMP/E installation process creates the ELM configuration utility libraries. When you run the REXX procedure for starting the configuration utility, the libraries are dynamically added to your TSO/ISPF session. The libraries are assigned only when needed 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 the following libraries:

hlq.SBLZEXEC
: REXX EXECs

hlq.SBLZLOAD
: Load modules

hlq.SBLZM*xxx*
: ISPF messages

hlq.SBLZP*xxx*
: ISPF panels

hlq.SBLZSLIB
: ISPF skeletons

hlq.SBLZSAMP
: Work flow XML

where *xxx* identifies the national language. For example, SBLZPENU is the ISPF panel library for U.S. English.

After you complete the installation of the configuration utility, refer to “Starting the configuration utility” on page 24 to start the utility and “Running the configuration utility” on page 25 for information about how to use the utility.

Chapter 2. Setting up the server on z/OS

Installing and configuring Jazz Team Server and the Engineering Lifecycle Management applications on z/OS systems

Installing the IBM Jazz Team Server and other IBM Engineering Lifecycle Management (ELM) 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.

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 IBM Engineering Lifecycle Management (ELM) products on z/OS, you must ensure that the SMP/E installation has ended successfully.

About this task

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 following parameter is affected:

- **MAXMMAPAREA** 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 Engineering Lifecycle Management applications

The Jazz Team Server (JTS) and each of the IBM Engineering Lifecycle Management (ELM) 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 ELM 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 9. Sample member names for each application installed on z/OS.

Installed application	Sample member
Change and Configuration Management (CCM) - HRCM703	BLZCPCCM
Global Configuration Management - HRGC703	BLZCPGC
IBM Jazz Reporting Service - HRRS703	BLZCPDCC, BLZCPRS, BLZCPLQE
Jazz Team Server (JTS) - HRJS703	BLZCPJTS
Link Index Provider - HRLI703	BLZCPLDX
Quality Management (QM) - HRQM703	BLZCPQM
IBM Engineering Lifecycle Optimization - Engineering Insights (ENI) - HRRE703	BLZCPRE

Table 9. Sample member names for each application installed on z/OS. (continued)	
Installed application	**Sample member**
Requirements Management (RM) - HRRM703	BLZCPRM
IBM WebSphere Liberty - HRWL703	BLZCPWLP

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), the Rational Build Agent (BLZCPBFA), and the Build Manager (BLZCPPVM) on z/OS. For instructions on running those members, see:

- "Creating additional Build System Toolkit directories" on page 59
- "Creating additional Rational Build Agent directories" on page 70

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

Table 10. 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/jazz703
@workPath@	The Jazz Team Server working directory.	BLZWORK	/u/jazz703

You must use unique directory names for @confPath@ and @workPath@ if you want to maintain additional configurations for previous versions.

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

@workPath@
6000 cylinders 3390, for running the Jazz Team Server and ELM applications on z/OS. This figure might be larger than required if the server is not running on z/OS or if the server is not running all of the ELM applications. Allocate a file system that allows for significant expansion as your installation grows.

@confPath@
300 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 ELM application directories require the following RACF file access permissions so the server can function:

1. The user ID that runs the IBM WebSphere Liberty server must have read and write access to @confPath@ and @workPath@.
2. The user ID that runs the Jazz Team Server repository tools are run must have read and write access to @confPath@ and @workPath@.

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 11. Additional variable names		
Variable Name	**Use**	**Default value**
BLZBFAH	The installation directory of the Rational Build Agent.	`/usr/lpp/jazz/v7.0.3/buildagent`
BLZBFAC	The Rational Build agent configuration files directory.	`/etc/jazz703/ccm`
BLZJAVA	The location of the Java directory.	`/usr/lpp/java/J11.0_64`
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 IBM Engineering Lifecycle Management (ELM) 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:

- Use ISPF option 3.17 to edit ASCII files under z/OS UNIX System Services.
- If you use IBM Developer for z/OS, 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 following files listed 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 IBM Engineering Lifecycle Management (ELM) applications that are installed.

Jazz Team Server (JTS)

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

```
@confPath@/jts/provision_profiles/nl2_profile.ini
@confPath@/jts/provision_profiles/nl2a_profile.ini
@confPath@/jts/provision_profiles/profile.ini
@confPath@/jts/provision_profiles/DISABLED/oidc-profile.ini
```

Change and Configuration Management (CCM)

```
@confPath@/ccm/provision_profiles/enterprise-update-site.ini
@confPath@/ccm/provision_profiles/nl1_enterprise-profile.ini
@confPath@/ccm/provision_profiles/nl1_profile.ini
@confPath@/ccm/provision_profiles/nl1_rtc-commons-profile.ini
@confPath@/ccm/provision_profiles/nl2_enterprise-profile.ini
@confPath@/ccm/provision_profiles/nl2_profile.ini
@confPath@/ccm/provision_profiles/nl2_rtc-commons-profile.ini
@confPath@/ccm/provision_profiles/nl2a_enterprise-profile.ini
@confPath@/ccm/provision_profiles/nl2a_profile.ini
@confPath@/ccm/provision_profiles/nl2a_rtc-commons-profile.ini
@confPath@/ccm/provision_profiles/profile.ini
@confPath@/ccm/provision_profiles/rtc-commons-profile.ini
@confPath@/ccm/provision_profiles/DISABLED/oidc-profile.ini
```

Quality Management (QM)

```
@confPath@/qm/provision_profiles/nl1_profile.ini
@confPath@/qm/provision_profiles/nl1_rtc-commons-profile.ini
@confPath@/qm/provision_profiles/nl2_profile.ini
@confPath@/qm/provision_profiles/nl2_rtc-commons-profile.ini
@confPath@/qm/provision_profiles/nl2a_profile.ini
@confPath@/qm/provision_profiles/nl2a_rtc-commons-profile.ini
@confPath@/qm/provision_profiles/profile.ini
@confPath@/qm/provision_profiles/rqm-nl1-profile.ini
@confPath@/qm/provision_profiles/rqm-nl2-profile.ini
@confPath@/qm/provision_profiles/rqm-nl2a-profile.ini
@confPath@/qm/provision_profiles/rqm-prereqs-profile.ini
@confPath@/qm/provision_profiles/rqm-profile.ini
@confPath@/qm/provision_profiles/DISABLED/oidc-profile.ini
```

Requirements Management - (RM)

```
@confPath@/rm/provision_profiles/fronting-server-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-fronting-server-web-profile.ini
@confPath@/rm/provision_profiles/nl1-rrcweb-profile.ini
@confPath@/rm/provision_profiles/nl2-fronting-server-profile.ini
@confPath@/rm/provision_profiles/nl2-fronting-server-web-profile.ini
@confPath@/rm/provision_profiles/nl2-rrcweb-profile.ini
@confPath@/rm/provision_profiles/nl2a-fronting-server-profile.ini
@confPath@/rm/provision_profiles/nl2a-fronting-server-web-profile.ini
@confPath@/rm/provision_profiles/nl2a-rrcweb-profile.ini
@confPath@/rm/provision_profiles/rrcweb-profile.ini
@confPath@/rm/provision_profiles/DISABLED/oidc-profile.ini
```

Global Configuration Management - (GC)

```
@confPath@/gc/provision_profiles/gc-profile.ini
@confPath@/gc/provision_profiles/nl1_gc_profile.ini
@confPath@/gc/provision_profiles/nl1_profile.ini
```

```
@confPath@/gc/provision_profiles/nl2_gc_profile.ini
@confPath@/gc/provision_profiles/nl2_profile.ini
@confPath@/gc/provision_profiles/nl2a_gc_profile.ini
@confPath@/gc/provision_profiles/nl2a_profile.ini
@confPath@/gc/provision_profiles/profile.ini
@confPath@/gc/provision_profiles/DISABLED/oidc-profile.ini
```

IBM Engineering Lifecycle Optimization - Engineering Insights - (ENI)

```
@confPath@/relm/provision_profiles/nl1-relm-repotools-profile.ini
@confPath@/relm/provision_profiles/nl1_profile.ini
@confPath@/relm/provision_profiles/nl1_relm_profile.ini
@confPath@/relm/provision_profiles/nl2-relm-repotools-profile.ini
@confPath@/relm/provision_profiles/nl2_profile.ini
@confPath@/relm/provision_profiles/nl2_relm_profile.ini
@confPath@/relm/provision_profiles/nl2a-relm-repotools-profile.ini
@confPath@/relm/provision_profiles/nl2a_profile.ini
@confPath@/relm/provision_profiles/nl2a_relm_profile.ini@confPath@/relm/
provision_profiles/profile.ini
@confPath@/relm/provision_profiles/relm-repotools-profile.ini
@confPath@/relm/provision_profiles/relm_profile.ini
@confPath@/relm/provision_profiles/DISABLED/oidc-profile.ini
```

Data Collection Component - (DCC)

```
@confPath@/dcc/provision_profiles/dcc_profile.ini
@confPath@/dcc/provision_profiles/license-profile.ini
@confPath@/dcc/provision_profiles/nl1-dcc_profile.ini
@confPath@/dcc/provision_profiles/nl1_profile.ini
@confPath@/dcc/provision_profiles/nl2-dcc_profile.ini
@confPath@/dcc/provision_profiles/nl2_profile.ini
@confPath@/dcc/provision_profiles/nl2a-dcc_profile.ini
@confPath@/dcc/provision_profiles/nl2a_profile.ini
@confPath@/dcc/provision_profiles/product-jfs-activation.ini
@confPath@/dcc/provision_profiles/profile.ini
@confPath@/dcc/provision_profiles/DISABLED/oidc-profile.ini
```

For example, if `@pathPrefix@` is set to *myroot*, then this setting resolves to `url=file:/`**myroot**`/usr/lpp/jazz/v7.0.3/server/conf/ccm/sites/update-site`.

Customizing the logging utility files

The `log4j2.xml` file configures the logging framework that is used by the Jazz Team Server and the other IBM Engineering Lifecycle Management (ELM) applications on z/OS systems.

A `log4j2.xml` file is included in each of the following application configuration directories, for each of the applications that you have installed and configured:

- `@confPath@/jts/log4j2.xml`
- `@confPath@/ccm/log4j2.xml`
- `@confPath@/qm/log4j2.xml`
- `@confPath@/rm/log4j2.xml`
- `@confPath@/gc/log4j2.xml`
- `@confPath@/relm/log4j2.xml`
- `@confPath@/dcc/log4j2.xml`

By default, the `log4j2.xml` files are configured to allow the Jazz Team Server and ELM applications to write messages to the log files specified in the `log4j2.xml` file. You can use information in these log files along with the information in the server job log files to identify a problem.

The `log4j2.xml` file is copied to the configuration directory specified by JAZZ_HOME and is updated automatically by the BLZCP* jobs. If you need to make more changes, edit the file in the application subdirectory under JAZZ_HOME, for example `/etc/jazz703/jts/log4j2.xml`.

Starting the configuration utility

Start the configuration utility using the **BLZCINIT** REXX executable code. You can run the executable code from the TSO command processor panel.

About this task

On the TSO command processor panel, enter `EX 'BLZ.SBLZEXEC(BLZCINIT)' 'BLZ BLZ.CONFIG'`. The ISPF client command **BLZCINIT** accepts three parameters:

HLQ
The data set name high level qualifiers for the configuration utility data sets. If this is not specified, the HLQ used to start **BLZCINIT** is used as the default.

USERHLQ
The data set name high level qualifiers for the configuration utility data sets created by the configuration utility process. If this is not specified, the default data set name high level qualifier is `BLZ.#CUST`.

Language
Identifies the national language. Currently, the configuration utility only supports the following languages:

ENU
U.S. English, which is the default if Language is omitted

ENP
U.S. English uppercase

JPN
Japanese

If either USERHLQ or Language invocation parameters need to be specified, then HLQ is required.

For example, run the **BLZCINIT** REXX executable code with the following command

```
ex 'BLZ.V703.SBLZEXEC(BLZCINIT)' 'BLZ.V703 BLZ.CONFIG ENU'
```

The resulting data set used are:

BLZ.V703.SBLZPENU and other BLZ.V703.*
System data sets

BLZ.CONFIG.ELMnnn.*
Configuration data sets

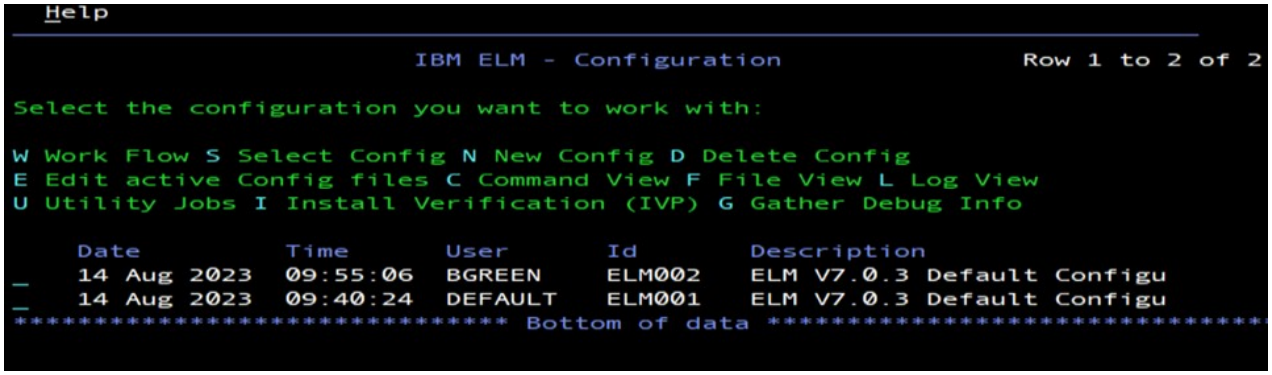
When the configuration utility is started, either for the first time or on subsequent invocations, the following information is presented to the user:

```
IBM ELM v7.0.3 utility initialization
HLQ for install datasets = BLZ.V703
HLQ for user datasets = BLZ.CONFIG
```

This information provides the version of the configuration utility, the product library high level qualifier used, and the high level qualifier of the configuration data sets.

What to do next

After starting the configuration utility for the first time, refer to "Running the configuration utility" at Running the configuration utility for information about how to use the utility to create a configuration. If you have already created a configuration, starting the configuration utility opens to the **Configuration** selection panel. You can enter actions to work with the configurations listed on the panel.



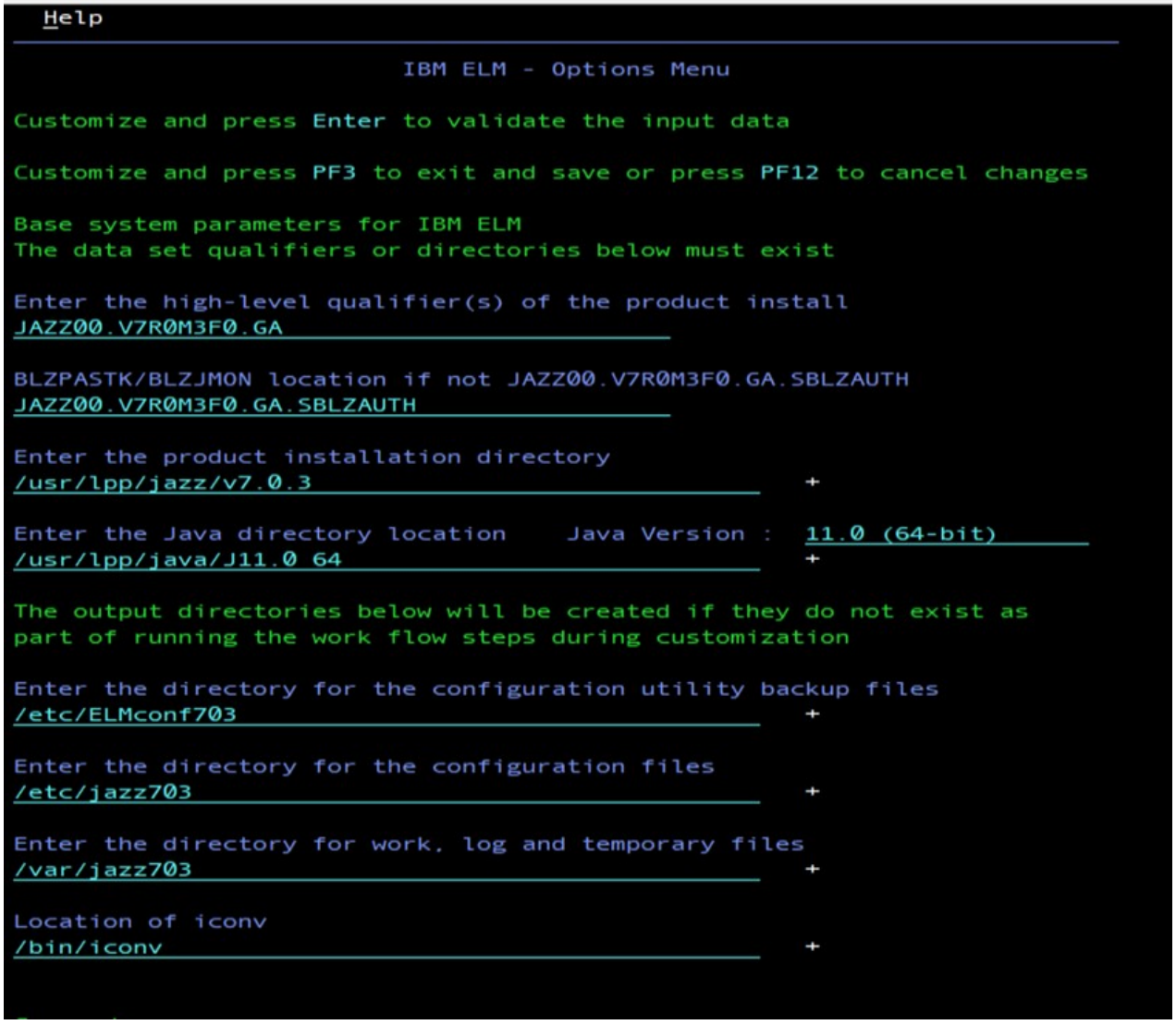
Running the configuration utility

When you start the configuration utility for the first time, or when you request a new configuration, you see the configuration utility **Options Menu**.

About this task

Enter the locations of the following information:

- The high-level qualifier (HLQ) of the installed product data sets. On initial start of the configuration utility, the HLQ defaults to the HLQ used to start the configuration utility.
- The location of the BLZPASTK module, which might have been moved to a secure library that is already APF authorized.
- The product installation directory in the hierarchical file system. By default, the installation directory is `/usr/lpp/jazz/v7.0.3`.
- The location of the Java code. The configuration utility uses the Java version found in your path, if one exists.
- The directory to be used for IBM Engineering Lifecycle Management (ELM) configuration utility backup files. By default, the directory is `/etc/ELMconf703`.
- The directory to be used for ELM configuration files. By default, the directory is `/etc/jazz703.`
- The directory to be used for work files, log files and temporary files. By default, the directory is `/u/jazz703`.
- The location of the `iconv` executable file. By default, `iconv` is located in `/bin/iconv`.



Change any option that is not correct and press the Enter key. You cannot continue to the next configuration utility panel until you enter a valid location for the product HLQ, product location, and Java location. If a location is specified that is not valid, an error screen is displayed.

When all information entered in the **Options Menu** is valid, the Java version is displayed.

```
Enter the Java directory location    Java Version :  11.0 (64-bit)
/usr/lpp/java/J11.0_64                               +
```

Press PF3 to proceed to the main **Customization** panel.

```
  Help
                          IBM ELM - Customization

                                                                    More:    -
Installation customization
   I Installation settings (*)
Required customization
   1 General settings (*)
   2 ISPF Gateway (*)

Optional tools customization
   3 Build Agent
   4 ISPF Client
   5 IDz Integration
   6 Job monitor
   7 Build Manager

Optional z/OS server customization
   8 WebSphere Liberty Profile

Optional EWM application customization
  9 Jazz Team Server (JTS)
  10 Change and Configuration Management (CCM)
  11 Quality Manager (QM)
  12 Requirements Management (RM)
  13 Global Configuration Management (GC)
  14 Engineering Insights (ENI)
  15 Link Index Provider (LDX)

  Jazz Reporting Services
  16 Lifecycle Query Engine (LQE)
  17 Report Builder (RS)
  18 Data Collection Component (DCC)

  19 Data Warehouse (DWH)

Select after the above configuration steps have been completed
   G Generate configuration jobs
```

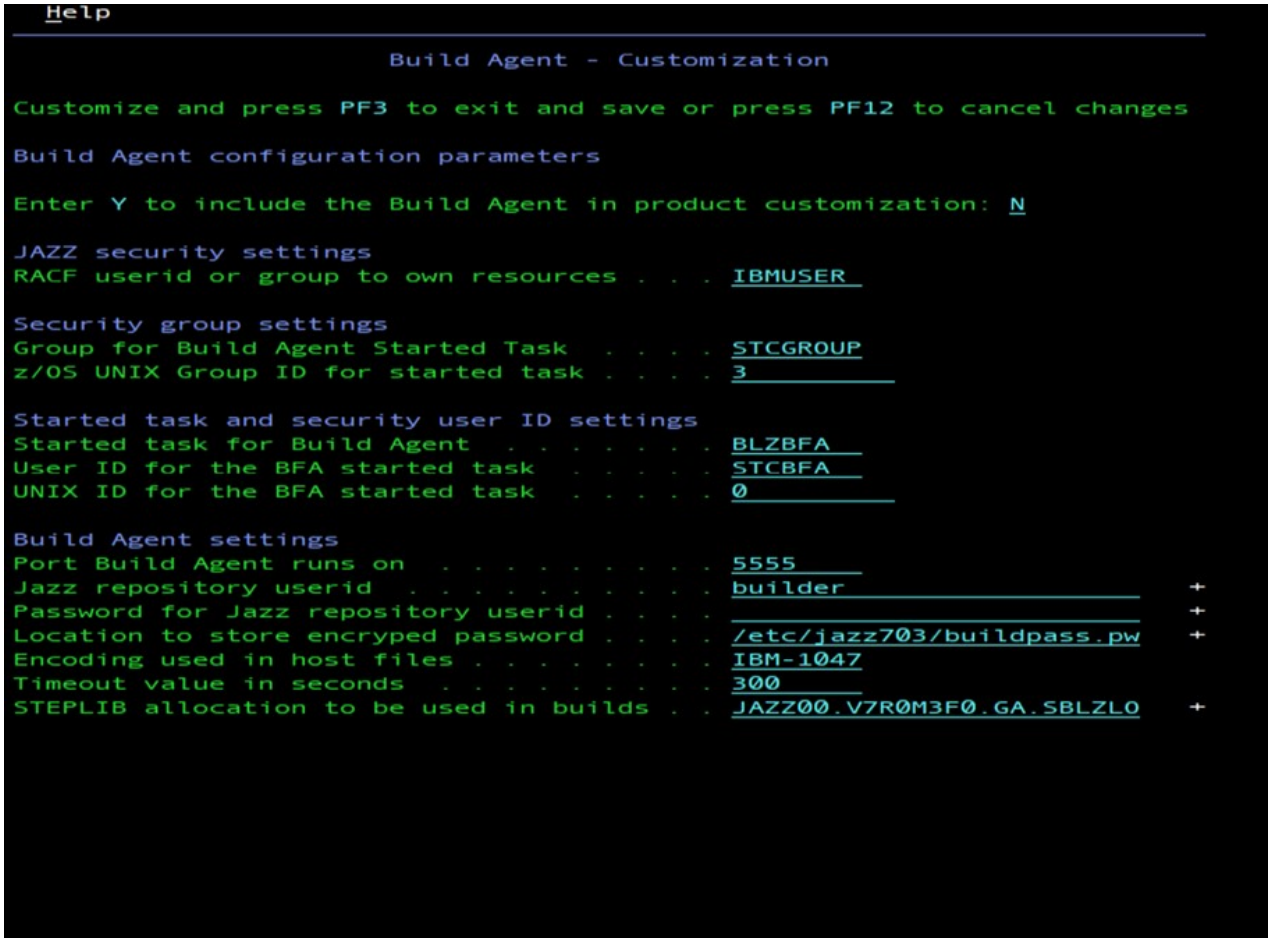
Select the components of the product you want to install. For example, if you are not running the Jazz Team Server on z/OS, you do not need to install any optional selections for the z/OS server or the IBM Engineering Workflow Management (EWM) application.

If you plan to use the Rational Build Agent to build z/OS applications but do not intend to use the ISPF client, you only need to review and configure the following options:

- **General settings (*)**
- **ISPF Gateway (*)**
- **Rational Build Agent**

An asterisk following an option indicates that the option is to be configured. The **General Settings** and **ISPF Gateway** options are required.

For any of the optional selections, select the option and specify in the customization screen that follows that you want to customize the optional component. For example, to install the Rational Build Agent, enter option 3 and press the Enter key to proceed to the **Rational Build Agent Customization** panel. Enter a Y to indicate that you want to include this component in the customization. In addition, review all of the default settings and change them, if necessary.



Select each component you want to include in the configuration, one at a time from the ELM configuration utility **Customization** panel and verify the settings in each panel. Press PF1 to open a help panel that provides more information about each setting on the individual **Customization** panels. In addition, more information can be found in the Interactive installation guide of the Engineering Workflow Management Documentation. After you generate a configuration work flow, you can modify the work flow by selecting and changing the configuration.

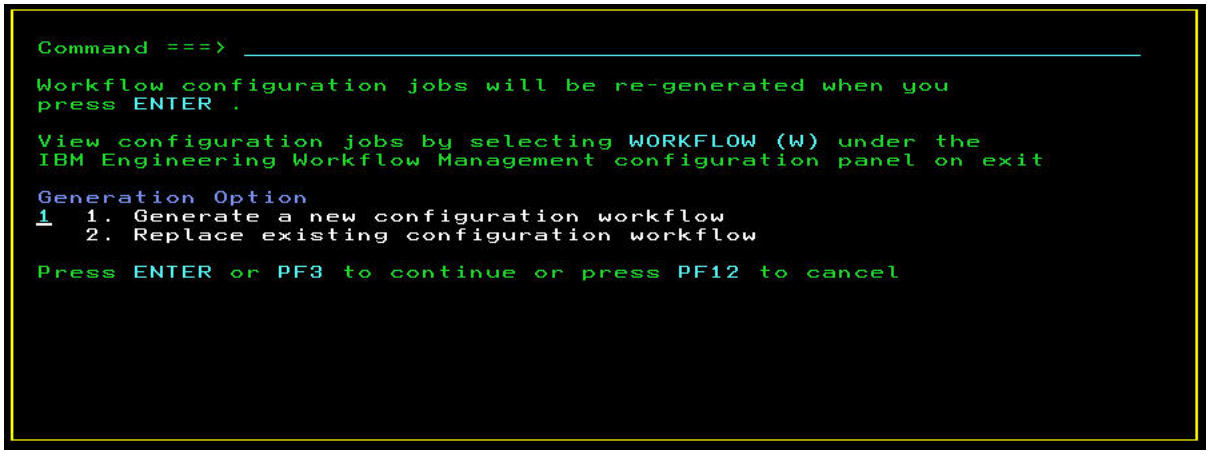
Generating a configuration workflow

When you finish checking the settings of the components you selected to configure, generate the work flow for the configuration you created.

About this task

On the configuration utility **Customization** panel, select option G to generate the configuration jobs and press Enter.

On the next panel, select whether to generate the configuration workflow for the configuration you created, or to replace an existing configuration workflow for a configuration you have edited, and press Enter.

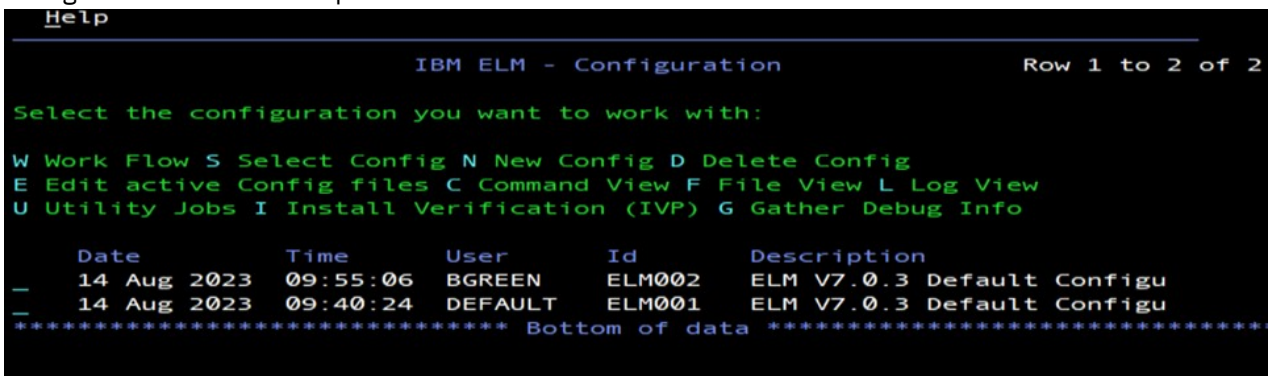


On the next panel, select one of the following options:

- Initialize a new workflow
- Save the status and log information from the current configuration

Press Enter to proceed to a panel on which you can provide a name for the generated configuration. When you have entered the name of the configuration, press Enter.

When the generation of the configuration workflow is complete, the configuration utility **Configuration** selection panel displays the new or changed configuration. You can enter actions to work with the configurations listed on the panel.

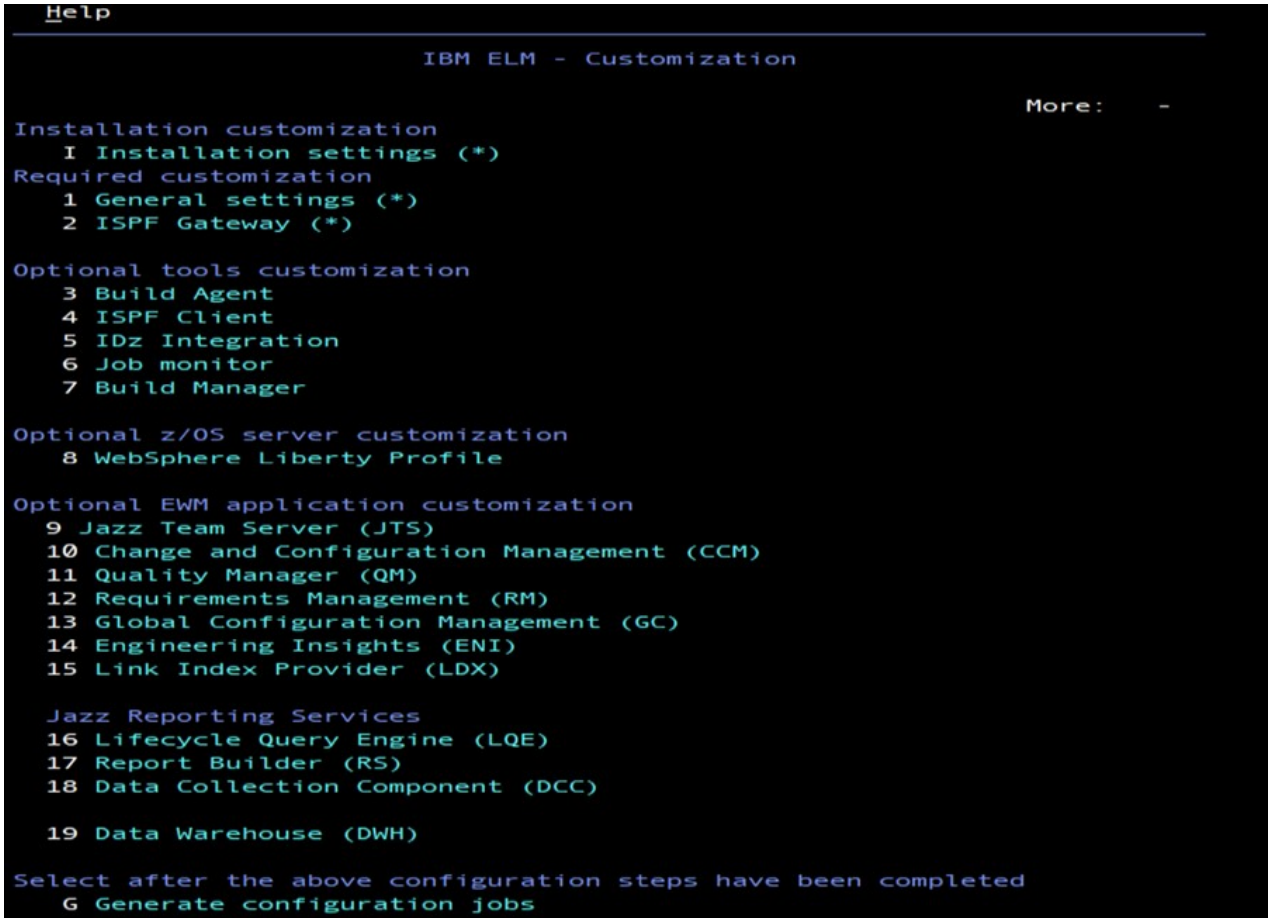


Modifying an existing configuration

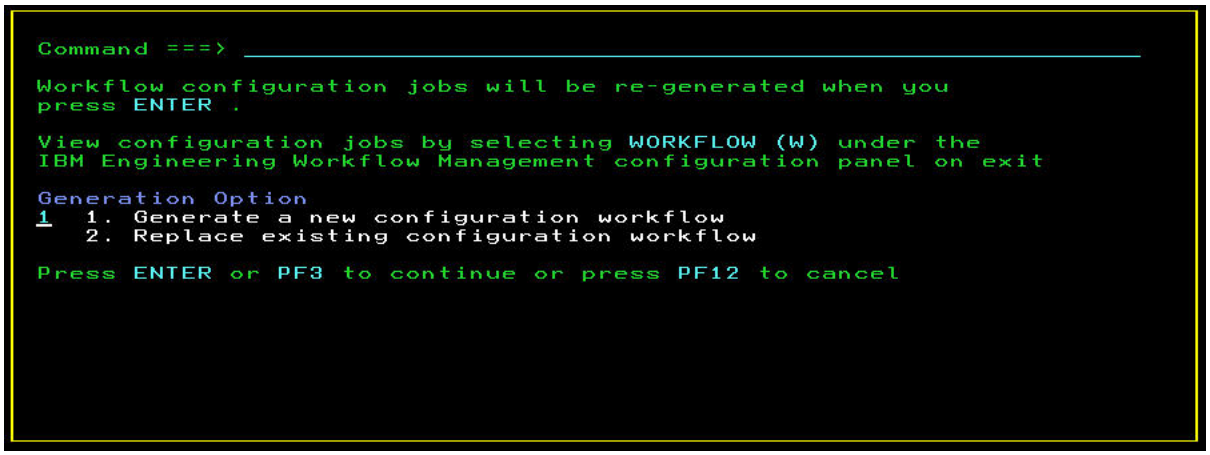
You can modify an existing configuration to change a parameter setting or to add a new optional component to the configuration.

About this task

On the configuration utility **Configuration** selection panel, enter option S next to the configuration you want to change and press Enter to display the main **Customization** panel.



Select the component that you want to modify. When you complete your modifications, enter option G to generate the workflow. On the next panel, select whether to generate the configuration workflow for the configuration you created, or to replace an existing configuration workflow for a configuration you have edited, and press Enter.



If you are adding a new component to an existing configuration or changing some variables, select option 2 to replace an existing configuration.

On the next panel, select one of the following options:

- Initialize a new workflow
- Save the status and log information from the current configuration

If you have already run some of the generated workflow execs, scripts or jobs, and you want to keep the logs and the current status of those workflow items, select option 2. For example, suppose that you had already run the workflow items related to the **General Settings** and the **ISPF Gateway**, but later added

the Rational Build Agent. You do not want to create a completely new configuration or lose any of the work you have already created. Option 2 saves the status and log information for the **General Settings** and the **ISPF Gateway**, and the Rational Build Agent workflow items are initialized as new workflow items.

Working with a configuration workflow

The components you chose to install determines the workflow items that are added to the workflow for a configuration.

About this task

To modify the configuration files and create databases and started tasks with your configuration, the workflow items need to be run. How the items are run depends on the type of workflow item. Enter a `W` next to the configuration you want to work with on the configuration utility **Configuration** selection panel. The **Work Flow** panel lists the workflow items generated by your configuration.

```
 Help
                                IBM ELM - Work Flow                 Row 1 to 14 of 14
Command ===> _                                                  Scroll ===> PAGE

The Work Items are listed in the suggested order of execution.
To Generate work items, select G - Generate configuration jobs
under the main ELM configuration panel.

Select the item you want to work with:

A Action Item E or S Edit V View C Mark as Completed L Browse Action Log

     Component       Work Item       Type            Status      Auth/Action
_    General Admin   RACFINIT        RACF            Pending     RACF Administra
_    General Admin   DATASET         RACF            Pending     RACF Administra
_    General Admin   USEROMVS        RACF            Pending     RACF Administra
_    General Admin   JAZZGRP         RACF            Pending     RACF Administra
_    General Admin   APF             PARMLIB         Pending     Systems Program
_    General Admin   BPXPRMXX        PARMLIB         Pending     Systems Program
_    -------------   -------------   ------------    ---------   ---------------
_    Build Toolkit   BLZCOPY         COMMAND         Pending     EWM Administrat
_    Build Toolkit   STARTISPF       startispf.sh    Pending     EWM Administrat
_    -------------   -------------   ------------    ---------   ---------------
_    ISPF Gateway    ISPZXENV        ISPZXENV        Pending     Systems Program
_    ISPF Gateway    ISPFCONF        ISPF.conf       Pending     Systems Program
_    ISPF Gateway    WORKAREA        COMMAND         Pending     System Programm
_    -------------   -------------   ------------    ---------   ---------------
********************************* Bottom of data ********************************
```

For a newly-created workflow, the status of all workflow items is set to **Pending**. The workflow items should be run in the order that they are listed. For some of the items, the order might not be important, but for other workflow items it is important. For example, Db2 tables must be created before the **`createTables`** job is run. You should follow the order of the workflow as much as possible. Select from the following list of actions for a workflow:

A - Action Item
Run the specified workflow item. The workflow item could be a RACF command, a REXX exec, or some SQL, to submit a job, or simply provide some instructions.

E or S - Edit
Edit the generated workflow item. The workflow item, which might be an exec, RACF command or Job, might need some local edits. If edits are made, they are saved so they can subsequently be run.

V - View
View the contents of the workflow item.

C - Mark as Completed
Some workflow items have a manual task that needs to be performed. Running the workflow item does not create anything. The manual task must be completed. When the manual task is completed, you can mark a workflow item as complete.

L - Browse Action Log
Browse the action log. This could be the output from a RACF command, a REXX exec or a job.

There are different types of workflow items, and the item type determines how the workflow item is run. The following list identifies the different types of workflow items:

RACF
A RACF command. If you run a RACF command, the workflow processor runs the RACF command. You can edit the workflow item before running it, for example, to replace the RACF statements with Top Secret or ACF2 statements.

PARMLIB
Update to a PARMLIB member. Running the workflow item does not create anything. The workflow item contains instructions about what you must do to complete the workflow item. Once you complete the instructions, you can mark the workflow item as complete.

PROCLIB
Update to a PROCLIB member. Running the workflow item does not create anything. The workflow item contains instructions about what you must do to complete the workflow item. Once you complete the instructions, you can mark the workflow item as complete.

COMMAND
Run a command. It might be a TSO command, a REXX exec, or an SQL command. The workflow processor runs the specified command and saves the log. The processor marks the status as complete when the command runs successfully or marks the status as failed if the command fails to run successfully.

JCL
Run a JCL member. The job is submitted when the workflow item is run. When the job is complete, the job output is retrieved and written to the log.

Note: For JCL workflow types, SDSF REXX is used to retrieve the job output. If your site does not use SDSF, use the provided user exit BLZCLMUX module to include your own code to retrieve the job output. If the call to SDSF REXX fails, then BLZCLMUX is called. You can edit the supplied user exit module that resides in the SBLZEXEC library to include code that retrieves the job output and writes it to the specified log member. Follow the workflow list and run each item following any additional instructions that might be presented to you. If the action fails, determine why; for example, a permission or space problem could exist. When the problem is resolved, issue the action command again. The new log replaces what was logged previously.

filename
The remaining workflow item types specify a file name such as `ISPZXENV` and `ISPF.conf`. Running these workflow items copies the generated file to the live configuration directory. Any existing information in the configuration directory is saved as backup.

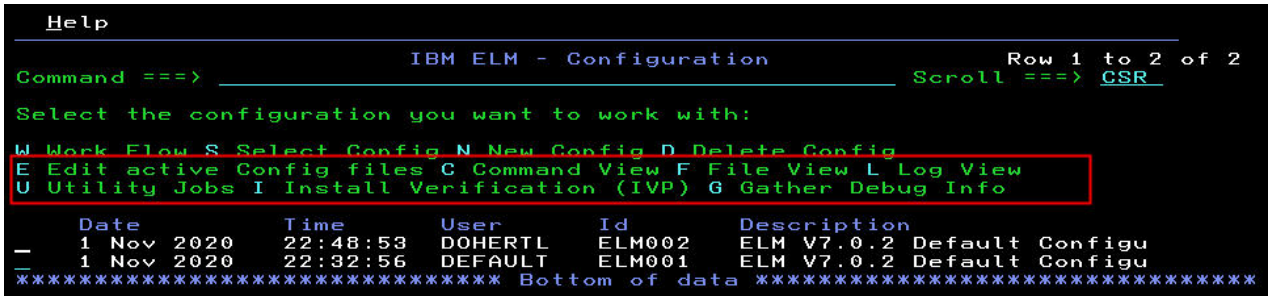
When all required workflow items have run successfully, you can use additional configuration options, available from the configuration utility **Configuration** selection panel.

Selecting additional configuration options

When all required workflow items have run successfully, you can specify additional configuration options.

About this task

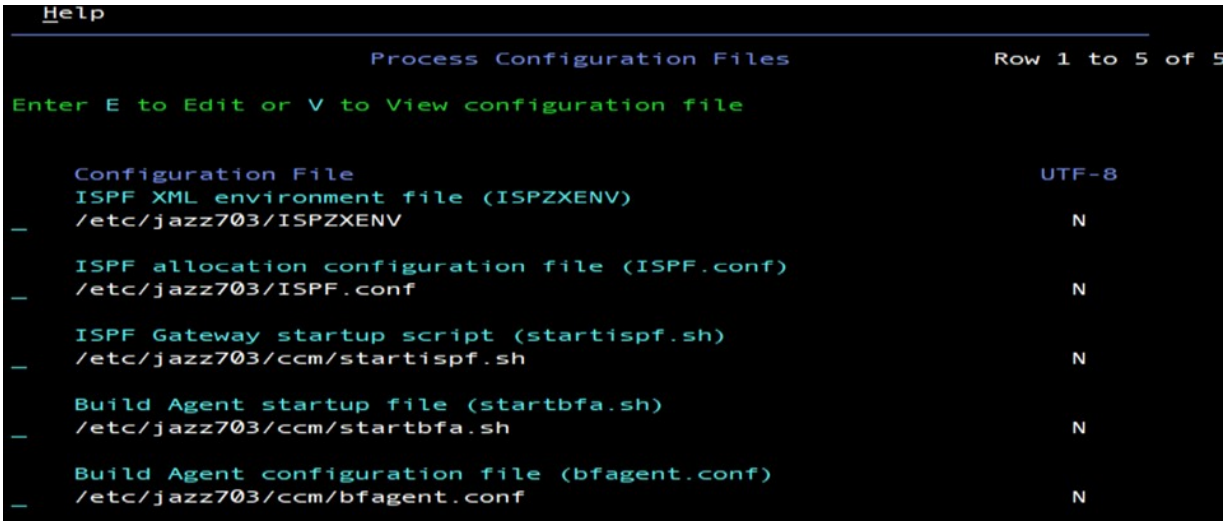
Select the additional configuration options from the configuration utility **Configuration** selection panel.



The following list describes the additional configuration options:

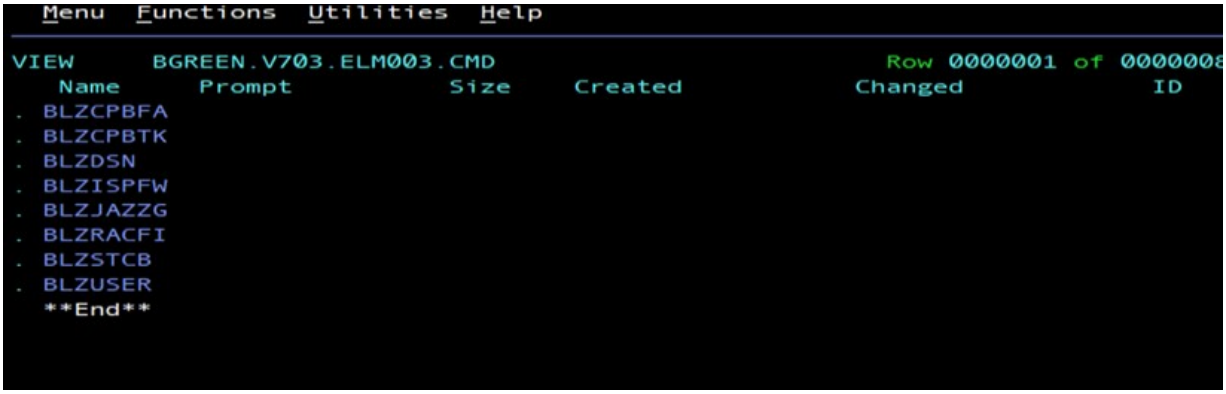
E - Edit active Config files

Make direct edits to the configuration files. For a particular configuration, all of the configuration files are listed in a single panel, which makes it easy to find and edit the files.



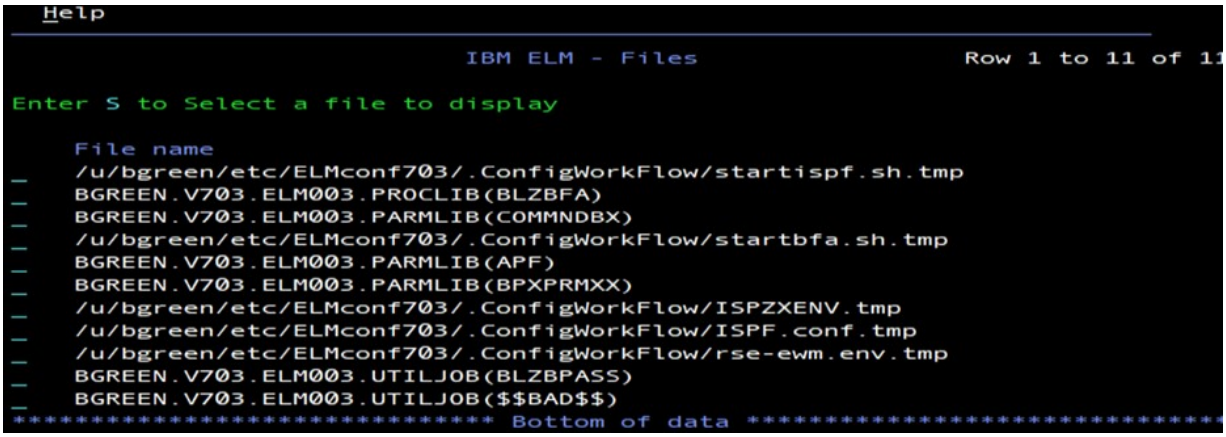
C - Command View

View the generated commands. You can also view the commands in the context of the workflow items from the **Work Flow** panel. However, this option shows a listing of the files in the data set in which they were created.



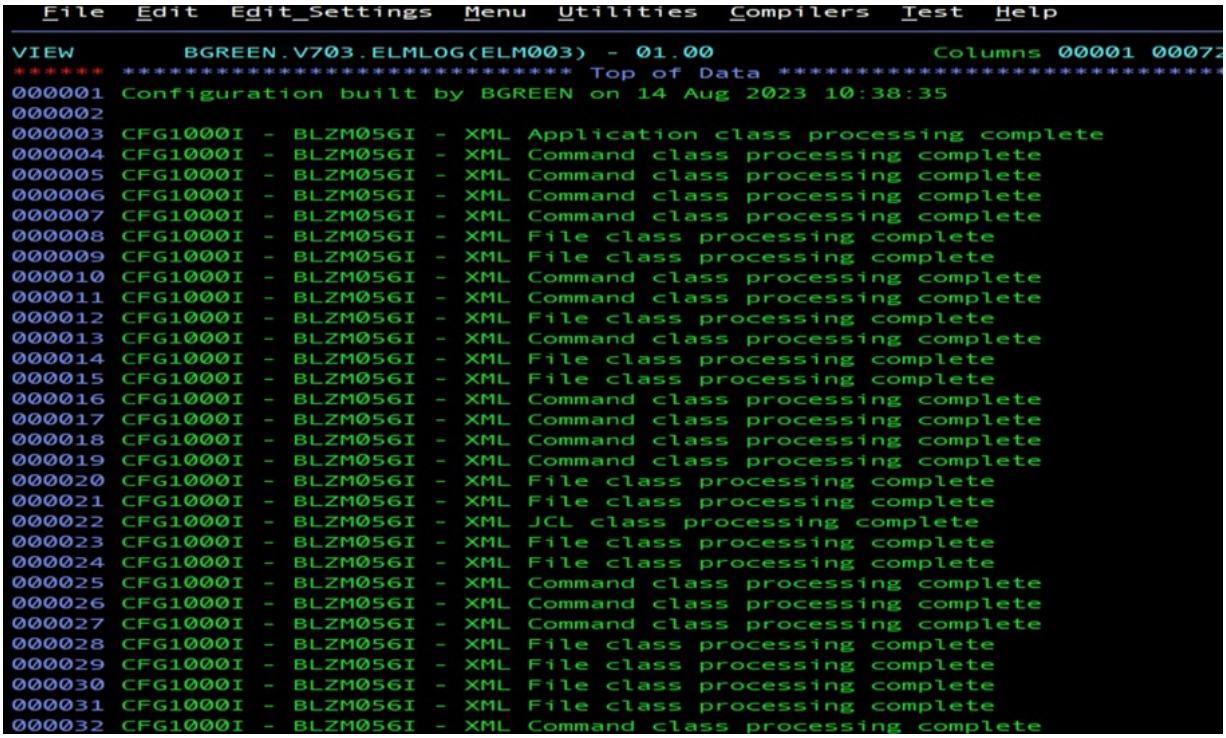
F - File View

View the generated files such as configuration files, PARMLIB members, and PROCLIB members. You can also view the commands in the context of the workflow items from the **Work Flow** panel. However, this option shows a listing of the files in the data sets and directories in which they were created.



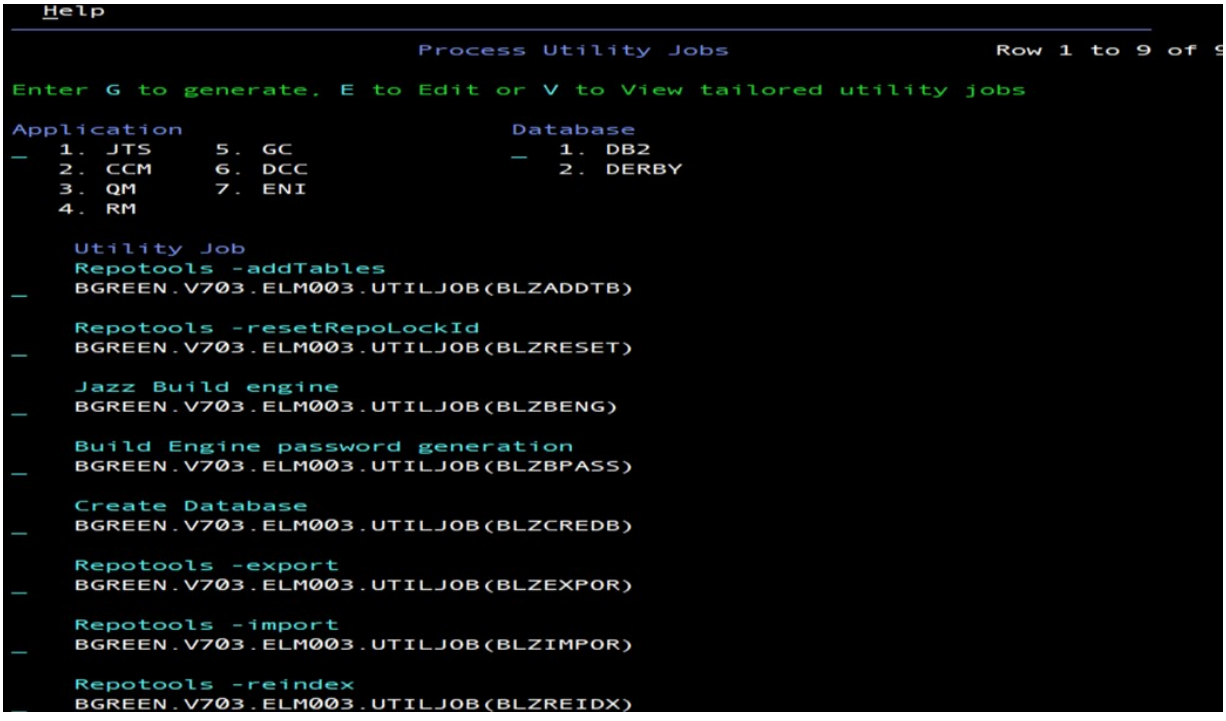
L - Log view

View the log. This is the overall log for a particular configuration rather than the specific logs relating to a workflow item.



U - Utility Jobs

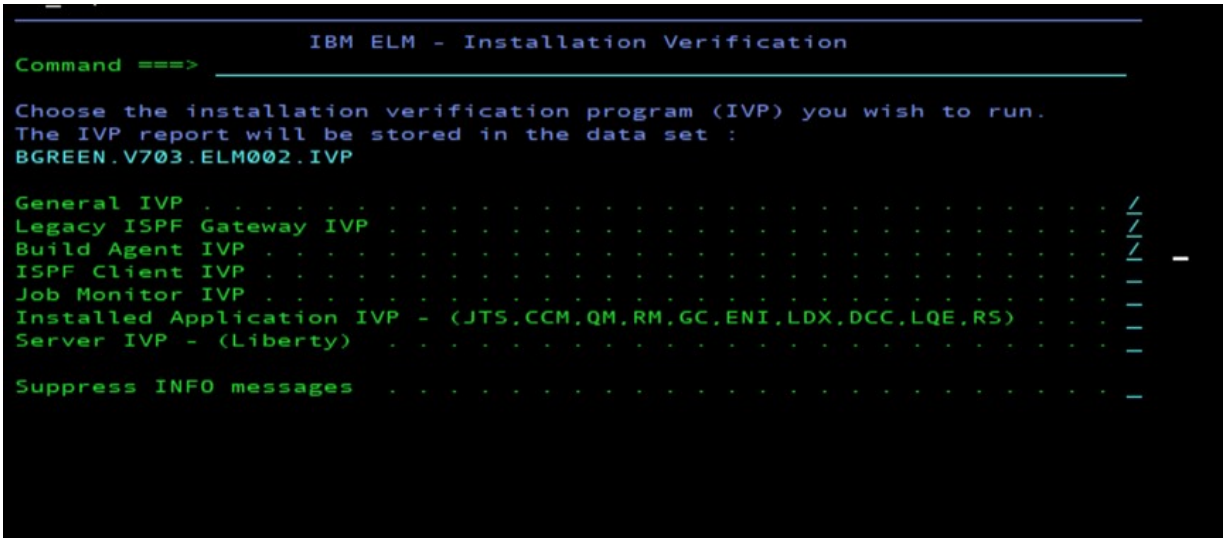
Generate utility jobs from templates. There are a number of jobs that are not part of a workflow, but are sometimes required, such as the **Repotools** jobs (like **Export**.) Tailor the jobs as needed from the options provided.



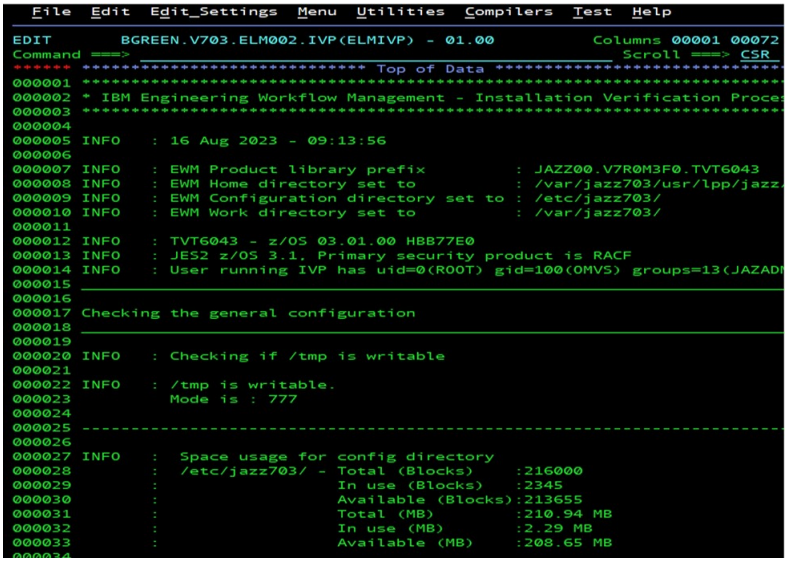
I - Installation Verification (IVP)

Run the installation verification process (IVP) in foreground mode. Select the parts of the IVP to run. You can suppress informational messages with Suppress INFO, so that only WARNING and ERROR messages are produced.

Tip: To ensure everything runs correctly, do not suppress the INFO messages the first time your run the IVP.



When the IVP has run, information about the installation is returned in a data set.



G - Gather Debug Info

When requested by IBM support, gather debug information. This option collect all of the logs, configuration files, generated commands, and other information that might be useful in debugging an installation problem.

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 Engineering Lifecycle Management applications on a z/OS system with IBM WebSphere Liberty on z/OS” on page 55.

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 for z/OS version 12.1 or 13.1 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
---------+---------+---------+---------+---------+---------+---
DSNWLM_GENERAL
DSNWLM_UTILS
```

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:

```
  IWM029I  10.30.01  WLM DISPLAY 124
    APPLICATION ENVIRONMENT NAME     STATE     STATE DATA
    BBOC001                          AVAILABLE
    BBOC002                          AVAILABLE
    BBOC003                          AVAILABLE
    BBOC004                          AVAILABLE
    DSNACCM0                         AVAILABLE
    DSNWLM_DEBUGGER                  AVAILABLE
    DSNWLM_DSNACICS                  AVAILABLE
    DSNWLM_GENERAL                   AVAILABLE
    DSNWLM_JAVA                      AVAILABLE
    DSNWLM_MQSERIES                  AVAILABLE
    DSNWLM_NUMTCB1                   AVAILABLE
    DSNWLM_PGM_CONTROL               AVAILABLE
    DSNWLM_REXX                      AVAILABLE
    DSNWLM_UTILS                     AVAILABLE
    DSNWLM_WEBSERVICES               AVAILABLE
    DSNWLM_XML                       AVAILABLE
```

In this case, you can see that the DSNWLM_GENERAL and DSNWLM_UTILS procedures are active.

Setting up Db2 for z/OS to use with Jazz Team Server

When running the Jazz Team Server and the IBM Engineering Lifecycle Management (ELM) applications with Db2 for z/OS, you must create a Db2 storage group and several Db2 databases, depending on which ELM 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 ELMSTG VOLUMES ('*') VCAT yourHlq ;
```

Notes:

1. The storage group can be named something other than *ELMSTG*.
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 ELM 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 JTS703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the CCM application
CREATE DATABASE CCM703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the QM application
CREATE DATABASE QM703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the RM application
CREATE DATABASE RM703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the LQE application
CREATE DATABASE LQE703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the DCC application
CREATE DATABASE DCC703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the GC application
CREATE DATABASE GC703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the LDX application
CREATE DATABASE LDX703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the ENI application
CREATE DATABASE ENI703 STOGROUP ELMSTG BUFFERPOOL BP16K0
 CCSID UNICODE;
 COMMIT;

Create the following Database for the Data Warehouse
CREATE DATABASE DW703 STOGROUP ELMSTG 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 or `com.ibm.team.datawarehouse.db.db2.dsn.dbname` 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 Engineering Lifecycle Management properties files for Db2 on z/OS" on page 39.
6. The reporting function in ELM 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.
7. You can specify database names and schema prefixes that are not tied to an ELM release so that when you upgrade, the names are still meaningful and when you clone or copy the repositories, the clone names can remain the same. If you clone a database, you must keep the schema prefix the same length, as it is required to run the ELM UNLOAD and LOAD utilities.

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, indexes, and views 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 ELMSTG 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. 32K buffer pool
-- is needed for inline LOBs
GRANT USE OF BUFFERPOOL BP16K0 TO jazz ;
GRANT USE OF BUFFERPOOL BE32K TO jazz ;

-- JTS – if the JTS repository will be on DB2 z/OS
GRANT DBADM ON DATABASE JTS703 TO jazz ;

-- CCM – if the CCM repository will be on DB2 z/OS
GRANT DBADM ON DATABASE CCM703 TO jazz ;

-- QM – if the QM repository will be on DB2 z/OS
GRANT DBADM ON DATABASE QM703 TO jazz ;

-- RM – if the RM repository will be on DB2 z/OS
GRANT DBADM ON DATABASE RM703 TO jazz ;--

-- LQE – if the LQE repository will be on DB2 z/OS
GRANT DBADM ON DATABASE LQE703 TO jazz ;--

-- DCC – if the DCC repository will be on DB2 z/OS
GRANT DBADM ON DATABASE DCC703 TO jazz ;--

-- GC – if the GC repository will be on DB2 z/OS
GRANT DBADM ON DATABASE GC703 TO jazz ;--

-- LDX – if the LDX repository will be on DB2 z/OS
GRANT DBADM ON DATABASE LDX703 TO jazz ;--

-- ENI – if the ENI repository will be on DB2 z/OS
```

```
GRANT DBADM ON DATABASE ENI703 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 DW703 TO jazz ;

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

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 43.

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 or change this setting.

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

If the value of field **DBADM CREATE AUTH** is set to **YES** on panel **DSNTIPP** during Db2 installation, you can create the database with **DBADM** authority, but if you want the user to upgrade or recreate the database, you must grant **SYSCTRL** or **SYSTEM DBADM** authorization to the user, because the user needs authority to be able to execute **DROP VIEW**.

```
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 defined in `teamserver.properties` using the property `com.ibm.team.datawarehouse.report.user`. 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 Engineering Lifecycle Management properties files for Db2 on z/OS

You can edit and configure the `teamserver.properties` files so that the server on z/OS can connect to Db2. The Jazz Team Server (JTS) and most of the other IBM Engineering Lifecycle Management (ELM) applications each have `teamserver.properties` files that contain server properties. The Lifecycle Query Engine (LQE) and Link Index Provider (LDX) have `dbconnection.properties` files that can be preconfigured. The Jazz Reporting Service (JRS) does not use Db2 on z/OS.

About this task

This topic describes how to configure the properties files. You can also input this configuration information into the Jazz Team Server setup wizard after starting the server.

Note: The Lifecycle Query Engine (LQE) and Link Index Provider (LDX) have `dbconnection.properties` files that can be preconfigured. The Jazz Reporting Service (JRS) does not use Db2 on z/OS.

Locate the configuration directories for each application that you are planning to deploy that uses teamserver.properties:

- Jazz Team Server (JTS) - `@confPath@/jts`
- Change and Configuration Management (CCM) - `@confPath@/ccm`
- Quality Management (QM) - `@confPath@/qm`
- Requirements Management (RM) - `@confPath@/rm`

- Data Collection Component (DCC) - *@confPath@*/dcc
- Global Configuration (GC) - *@confPath@*/gc
- IBM Engineering Lifecycle Optimization - Engineering Insights (ENI) - *@confPath@*/relm

On z/OS, the *@confPath@* is the path value that was used when customizing the server using the BLZCP* sample jobs, such as /etc/jazz703. 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:

```
DSNL080I  -DBC1 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION           LUNAME            GENERICLU
DSNL083I DBC1               NET1.DBC1LU       -NONE
DSNL084I TCPPORT=5050  SECPORT=5052     RESPORT=5051  IPNAME=-NONE
DSNL085I IPADDR=9.30.243.101
DSNL086I SQL    DOMAIN=TVT6012.svl.ibm.com
DSNL105I CURRENT DDF OPTIONS ARE:
DSNL106I PKGREL = COMMIT
DSNL106I SESSIDLE = 001440
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 ten repositories: one repository for JTS, one repository for each installed component that has a database (CCM, QM, RM, LQE, DCC, GC, LDX, ENI), 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.
- You can specify database names and schema prefixes that are not tied to an ELM release so that when you upgrade, the names are still meaningful and when you clone or copy the repositories, the clone names can remain the same. If you clone a database, you must keep the schema prefix the same length, as it is required to run the ELM UNLOAD and LOAD utilities.

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 ELM 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 JTS703_REPOSITORY, in a given database when `com.ibm.team.repository.db.schemaPrefix=`**JTS**703. 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=DW703
#com.ibm.team.datawarehouse.db.schemaPrefix=DW703X
```

   You must add the following property, if it does not exist, to specify a prefix of your choice for the data warehouse tables:

```
com.ibm.team.datawarehouse.db.schemaPrefix=DW703X
```

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 DW703.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`

If you choose to deploy the LQE or LDX applications, you can configure the connection data or provide the data as part of the Jazz Team Server setup wizard. Alternatively, to preconfigure the applications, create a file named dbconnection.properties in the configuration directories for the applications. The format of the file is similar to the `teamserver.properties` file, but includes only the following lines:

```
# DB2z Configuration. Please configure the following properties
db.vendor=DB2Z
db.location=//@hostname@:@port@/@location@:user=@username@;password={password};
db.password=@password@
db.name=@dbname@
db.schema.prefix=@schema@
```

Substitute the variables that have the @ symbols on each side of the variable, such as `db.name=LQE703`. The database tables are created when the server is started for the LQE and LDX applications, not by repository tools commands.

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 Redbooks® publication LOBS with DB2 for z/OS: Stronger and Faster.

Configuring a Jazz Team Server that is not on z/OS to use a Db2 repository on z/OS

You can configure a Jazz Team Server that is not installed on a z/OS system to connect to a Db2 repository that is installed on a z/OS system.

About this task

These instructions describe how to connect to Db2 on z/OS and are intended for database administrators. The instructions apply to all platforms. Complete the tasks required for setting up a Db2 database on z/OS, beginning with the following topic: “Setting up a Db2 database on a z/OS system” on page 35.

Complete the steps in the following procedure to configure a Jazz Team Server that is not on z/OS to use a Db2 repository on z/OS.

Procedure

If you are using a Db2 database on z/OS, the non-z/OS server still requires access to JDBC license `.jar` files that are shipped with Db2 for z/OS. To provide access to the license files, complete the following steps:

1. Create a directory called `db2z` within the `<JazzInstallDir>/server`.
2. In the `db2z` directory you created, save copies of the `db2jcc_license_cisuz.jar` and `db2jcc4.jar` files from your z/OS system.

   Note:

   - For Db2 version 12.1, you can find the files in the Db2 12.1 SDSNJCC file system, which is mounted at a file path like: `/usr/lpp/db2c10/jdbc/classes`.
   - For Db2 version 13.1, you can find the files in the Db2 13.1 SDSNJCC file system, which is mounted at a file path like: `/usr/lpp/db2d10/jdbc/classes`.

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 IBM Engineering Lifecycle Management (ELM) 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 Engineering Lifecycle Management properties files for Db2 on z/OS” on page 39 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 V12 or V13 JDBC license and JDBC .jar files on your system. These files are named `db2jcc_license_cisuz.jar` and `db2jcc4.jar` and are located by default at `/usr/lpp/db2c10/jdbc/classes` for V12, and `/usr/lpp/db2d10/jdbc/classes` for V13.
- 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`
   - RM tables (if required) using `application = rm` and `repotoolsCommand = -createTables`
   - DCC tables (if required) using `application = dcc` and `repotoolsCommand = -createTables`
   - GC tables (if required) using `application = gc` and `repotoolsCommand = -createTables`
   - ENI tables (if required) using `application = eni` 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.

   The Lifecycle Query Engine (LQE) and Link Index Provider (LDX) repositories can use Db2 z/OS, but these applications do not use repository tools to create database tables. Tables are created when the server is started.

   Report Builder (RS) data is not stored in Db2 z/OS but is stored in the filesystem, and also does not use repository tools.

   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 ;

-- Grants for RM assuming schema prefix of RMX and user ID of jazz
GRANT DELETE, INSERT, SELECT, UPDATE
```

```
    ON TABLE
         RMX_COMMON_SNAPSHOT.SMPL_ITER,
         RMX_REPOSITORY.CONTRIBUTOR,
         RMX_REPOSITORY_SNAPSHOT.LATEST
       TO JAZZ ;

-- Grants for DCC assuming schema prefix of DCCX and user ID of jazz
GRANT DELETE, INSERT, SELECT, UPDATE
    ON TABLE
         DCCX_COMMON_SNAPSHOT.SMPL_ITER,
         DCCX_REPOSITORY.CONTRIBUTOR,
         DCCX_REPOSITORY_SNAPSHOT.LATEST
       TO JAZZ ;

-- Grants for GC assuming schema prefix of GCX and user ID of jazz
GRANT DELETE, INSERT, SELECT, UPDATE
    ON TABLE
         GCX_COMMON_SNAPSHOT.SMPL_ITER,
         GCX_REPOSITORY.CONTRIBUTOR,
         GCX_REPOSITORY_SNAPSHOT.LATEST
       TO JAZZ ;
-- Grants for ENI assuming schema prefix of ENIX and user ID of jazz
GRANT DELETE, INSERT, SELECT, UPDATE
    ON TABLE
         ENIX_COMMON_SNAPSHOT.SMPL_ITER,
         ENIX_REPOSITORY.CONTRIBUTOR,
         ENIX_REPOSITORY_SNAPSHOT.LATEST
       TO JAZZ ;
```

Creating Db2 for z/OS database tables remotely

You can set up servers to run on a non-z/OS system that accesses Db2 database tables on z/OS.

About this task

You can create the database tables remotely if you meet the following conditions:

- You have customized the `teamserver.properties` for Db2 for z/OS. See “Customizing the Jazz Team Server and Engineering Lifecycle Management properties files for Db2 on z/OS” on page 39.
- You have completed the additional configuration steps to make the Db2 for z/OS JDBC and license .jar files available on your system. See “Configuring a Jazz Team Server that is not on z/OS to use a Db2 repository on z/OS” on page 43.

  Important: Run the **`createTables`** command only if this is a new installation.

  Note:

  – When you run the **`repotools`** command, the `teamserver.properties` file will be updated and the original file with the original content will be saved in the same directory.
  – You might see a warning that the public URI has not been set. You can ignore this warning at this time because the public URI will be set when you run the setup wizard. For details, see Running the setup by using Custom setup in the setup wizard in the Engineering Lifecycle Management Documentation.

  The Lifecycle Query Engine (LQE) and Link Index Provider (LDX) repositories can use Db2 z/OS, but these applications do not use repository tools to create database tables. Tables are created when the server is started.

  Report Builder (RS) data is not stored in Db2 z/OS but is stored in the filesystem, and also does not use repository tools.

Procedure

1. For the Jazz Team Server, go to the *`JazzInstallDir`*`/server` directory and run the following commands:

   ```
   repotools-jts -createTables
   repotools-jts -createWarehouse
   ```

2. For the Change and Configuration Management (CCM) application, go to the *JazzInstallDir*/server directory and run the following command:

```
repotools-ccm -createTables
```

3. For the Quality Management (QM) application, go to the *JazzInstallDir*/server directory and run the following command:

```
repotools-qm -createTables
```

4. For the Requirements Management (RM) application, go to the *JazzInstallDir*/server directory and run the following command:

```
repotools-rm -createTables
```

5. For the Data Collection Component (DCC) application, go to the *JazzInstallDir*/server directory and run the following command:

```
repotools-dcc -createTables
```

6. For the Global Configuration Management (GC) application, go to the *JazzInstallDir*/server directory and run the following command:

```
repotools-gc -createTables
```

7. For the IBM Engineering Lifecycle Optimization - Engineering Insights (ENI) application, go to the *JazzInstallDir*/server directory and run the following command:

```
repotools-relm -createTables
```

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 ;
-- Grants for RM assuming schema prefix of RMX and user ID of jazz
GRANT DELETE, INSERT, SELECT, UPDATE
    ON TABLE
         RMX_COMMON_SNAPSHOT.SMPL_ITER,
         RMX_REPOSITORY.CONTRIBUTOR,
         RMX_REPOSITORY_SNAPSHOT.LATEST
      TO JAZZ ;

-- Grants for DCC assuming schema prefix of DCCX and user ID of jazz
GRANT DELETE, INSERT, SELECT, UPDATE
    ON TABLE
         DCCX_COMMON_SNAPSHOT.SMPL_ITER,
         DCCX_REPOSITORY.CONTRIBUTOR,
         DCCX_REPOSITORY_SNAPSHOT.LATEST
       TO JAZZ ;
```

```
-- Grants for GC assuming schema prefix of GCX and user ID of jazz
GRANT DELETE, INSERT, SELECT, UPDATE
    ON TABLE
         GCX_COMMON_SNAPSHOT.SMPL_ITER,
         GCX_REPOSITORY.CONTRIBUTOR,
         GCX_REPOSITORY_SNAPSHOT.LATEST
       TO JAZZ ;

-- Grants for ENI assuming schema prefix of ENIX and user ID of jazz
GRANT DELETE, INSERT, SELECT, UPDATE
    ON TABLE
         ENIX_COMMON_SNAPSHOT.SMPL_ITER,
         ENIX_REPOSITORY.CONTRIBUTOR,
         ENIX_REPOSITORY_SNAPSHOT.LATEST
       TO JAZZ ;
```

Managing Db2 for z/OS databases before upgrading IBM Engineering Lifecycle Management

Each new Engineering Lifecycle Management (ELM) version or release updates the data model to change the database schema for Jazz Team Server and the ELM applications by using the repository tools **addTables** command. The upgrade process tags the repository with a version and release number, which means that it can be used only with that version and release. You must ensure that you have adequate backups for the tables in case there is an error and you must roll back the data. In addition to any tools you might have, IBM also provides tools for using Db2 for z/OS **UNLOAD** and **LOAD** to create a backup copy. It is important to understand that just generating the DDL from the existing tables and loading the current data into them might not be sufficient, as the repository tools **-createTables** process creates the tables, inserts data into some tables, and establishes links between tables. Table owner names might be different between instances of the database.

About this task

It is critical that the backup and restore process for the data is understood and tested before any upgrade. Testing the backup and restore process also creates a set of duplicate tables that can be used to test the upgrade process itself, preferably in a separate subsystem.

The upgrade cannot be rolled back without a repository backup.

Procedure

1. To back up your ELM database with the Db2 for z/OS utilities, you can use the created backup database to restore your database if you have any problems during an upgrade from one version of ELM to another.
   a) Unload the database and use the unloaded data sets to create a backup database.
      i) Run the UNLOAD process. To do so, follow the instructions in “Running UNLOAD and LOAD processes on Db2 to create a backup database” on page 50.
      ii) Ensure that you meet your database prerequisites.

      You must have the correct user name and password and your user ID must have correct Db2 for z/OS authority. Test the **addTables** process on a system with an ID that has the same authority and is configured in the same way from a Db2 perspective. The DBADM authority is not sufficient because the **addTables** process runs **DROP VIEW**, which requires additional authority. See “Setting up Db2 for z/OS to use with Jazz Team Server” on page 36.
   b) Back up any other Db2 or third-party programs, if necessary.

      Important: Use the programs and JCL samples that are provided in the HRCC703 FMID to back up all the required programs. If you do not, some required programs might be missing from the backup.
2. To restore your ELM database with the Db2 for z/OS utilities, complete the following tasks:
   a) Reload the database to a different Db2 subsystem or a different database name and schema prefix.

Run the LOAD process to reload a new or existing set of Db2 for z/OS tables. To do so, follow the instructions in "Running UNLOAD and LOAD processes on Db2 to create a backup database" on page 50.

b) Restore any other Db2 or third-party programs, if necessary.

Important: Use the JCLs that are provided in the HRCC703 FMID to restore all the required programs. If you do not, some required programs might be missing.

Results

You are now ready to upgrade the ELM version.

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 IBM Engineering Lifecycle Management (ELM) 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 Engineering Lifecycle Management 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 Engineering Lifecycle Management repositories.

These programs are compatible with the current and previous versions of IBM Engineering Workflow Management (EWM) in Engineering Lifecycle Management. You can only unload and load database tables for the same version of the product. It is recommended you use the latest available version of these programs.

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. 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 must 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.

If you create tables for a target database that is different from the current database, use a schema prefix that has the same length, as it is required by the UNLOAD and LOAD utilities.

Shipped components to aid backup and recovery

The following components are included in the SMP/E package in FMID HRCC703.

BLZUNLD
A program in *hlq*.SBLZL0AD 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 ELM Db2 for z/OS database (sample JCL).

BLZDB2ED
Sample REXX member in *hlq*.SBLZSAMP that edits the generated unload statements for LOB tablespaces to remove the DB2_GENERATED_ROWID_FOR_LOBS statement prior to running the unload of the LOB tablespaces.

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 ELM database tables.

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

BLZDWHLD
Sample JCL member in *hlq*.SBLZSAMP that is used to load the ELM 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.

Considerations before running unload/load JCL

If your tables have large tablespaces, you might need to modify the JCL provided in sample members BLZDB2UN, BLZDB2LD, BLZDWHUN and BLZDWHLD to temporarily allocate more space for the tables. For example, in the Db2 Load procedures, you might need to modify BLZDB2LD and BLZDWHLD to change the space allocations of the SYSUT1, SYSMAP, SORTOUT DDs and the SORTWK* allocations.

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 ELM tables (except the data warehouse tables)
Sample jobs are provided that can be used to unload the tables of Jazz Team Server (JTS), of Change and Configuration Management (CCM), and of the other applications.

Note: RM UNLOAD/LOAD is supported starting with V5.0.

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. Call program BLZDB2ED to edit the generated unload statements, for LOB tablespaces, to remove the DB2_GENERATED_ROWID_FOR_LOBS statements.

3. Print the UNLOAD utility control statements for reference.
4. Unload the 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.

Complete the following steps to unload your ELM tables:

1. Stop the server.
2. If any of your ELM tables are in Db2 for z/OS, modify and submit the BLZDB2UN job in *hlq*.SBLZSAMP to unload the corresponding 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 ELM 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. In addition, the BLZSYSIN program removes the load of the PACKAGE_MAP and TABLE_MAP tables from each schema. These tables are created by the **-createTables** process. If the PACKAGE_MAP and TABLE_MAP tables are loaded from a different installation, they might cause problems.
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 ELM 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 JTS703;
COMMIT;

Create the following Database for the Jazz Team Server
CREATE DATABASE JTS703 STOGROUP ELMSTG 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 36.
3. Create the new ELM tables by running a **repotools -createTables** command for each 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 tables of JTS, CCM, and the other applications by changing the **@appl@** parameter as described in the instructions included with the jobs.

   For more information about running this job, see "Creating database tables using repository tools" on page 43.

4. Modify and submit the BLZDB2LD job in *hlq*.SBLZSAMP to load the ELM 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 ELM 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 ELM tables:

1. Stop the server.
2. Create the necessary tables as documented in "Setting up Db2 for z/OS to use with Jazz Team Server" on page 36. 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 ELM tables by running a **`repotools -createTables`** command for each 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 tables of JTS, CCM, and the other applications by changing the **`@appl@`** parameter as described in the instructions included with the jobs.

   For more information about running this job, see "Creating database tables using repository tools" on page 43.
4. Modify and submit the BLZDB2LD job in *hlq*.SBLZSAMP to load the ELM 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 ELM 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 ELM 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 ELM 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 DW703;
COMMIT;

Create the following Database for the Jazz Team Server
CREATE DATABASE DW703 STOGROUP ELMSTG 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 36.
3. Create the new ELM 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 about running this job, see “Creating database tables using repository tools” on page 43.
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 ELM 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 36.

   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 ELM 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 about running this job, see "Creating database tables using repository tools" on page 43.
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 "Running Jazz Team Server and the Engineering Lifecycle Management applications on a z/OS system with IBM WebSphere Liberty on z/OS" on page 55.

Running Jazz Team Server and the Engineering Lifecycle Management applications on a z/OS system with IBM WebSphere Liberty on z/OS

Running the Jazz Team Server and the IBM Engineering Lifecycle Management (ELM) applications on a z/OS system with WebSphere Liberty on z/OS involves setting up security and configuring the WebSphere Liberty.

About this task

These instructions supplement the basic considerations for deploying and starting WebSphere Liberty on z/OS.

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 in the IBM Engineering Lifecycle Management Documentation.

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 IBM Engineering Lifecycle Management (ELM) applications are configured to use RACF for security, the IBM WebSphere Liberty server determines repository permissions based on RACF EJBROLE profiles for security control.

Define the Jazz Team Server and ELM 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.

   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 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 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, JazzProjectAdmins, JazzGuests, JazzUsers).
   - For IBM WebSphere Liberty:
     - A System Authorization Facility (SAF) profile prefix is specified in the `server.xml` and `jvm.options` files that affects the way the EJBROLE profiles are referenced. For more information about using EJBROLE profiles, search the WebSphere Liberty Server for z/OS for the version that you are running for the *System Authorization Facility for role-based authorization* topic.
     - You must grant READ access to the APPL profile for the Engineering Workflow Management users. Search the WebSphere Liberty Server for z/OS for the version that you are running for more information.
     - The value for `profilePrefix` in `server.xml` 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 IBM Developer for z/OS.

Configuring Jazz Team Server and the applications for the Engineering Lifecycle Management to use IBM WebSphere Liberty on z/OS

This topic describes how to set up the Jazz Team Server and the IBM Engineering Lifecycle Management (ELM) applications to work with the IBM WebSphere Liberty on z/OS.

Before you begin

Before you perform this task, make sure you:

- Used SMP/E to install HRWL703 and other necessary FMIDs based on the applications you will be configuring.
- Customized and submitted BLZCPWLP to create the IBM WebSphere Liberty server and configuration files.
- Customized and submitted the BLZCP* jobs for the Jazz Team Server and any ELM applications you plan to deploy.
- Created the Db2 databases and updated the `teamserver.properties` files with your database settings.
- Established RACF security settings. For more information about RACF security, see “Security on z/OS systems” on page 9 and “Setting up user security on a z/OS system by using RACF” on page 55.
- Verified or updated the IBM WebSphere Liberty level. Jazz Team Server requires IBM WebSphere Liberty on z/OS v19.0.0.6 or higher.

About this task

Refer to the WebSphere Liberty Server for z/OS. Select the version that you are running, and search for “Setting up user security on a z/OS system by using RACF” on page 55.

Procedure

Perform the following steps to deploy the Jazz Team Server and ELM applications on the IBM WebSphere Liberty. Each step is detailed in its own task.

1. Review and update server properties files.
2. Copy server started task procedures to PROCLIB and modify them.
3. Start the application server processes.

Review and update server properties files

About this task

The BLZCPWLP sample job creates an IBM WebSphere Liberty and creates initial configuration files based on the provided values. Follow the instructions in this section to review these files and to make additional changes, if needed to match your preferred configuration.

The `server.xml` configuration file can not be changed; it is a *signed* file provided by IBM. If you think you need changes to the *signed* configuration file, contact IBM Support.

Procedure

1. Locate the initial configuration files. By default, these files are created in the `/u/jazz703/wlpuser/servers/clmserver` directory, but the location is determined by the values you used when you ran BLZCPWLP.

2. Review and update the following files, if necessary:

   bootstrap.properties
   You can use this file to control logging and tracing. The log directory should be correct. Adjust other settings as necessary.

   jvm.options
   You can modify any of the settings in this file if they are not correct for your configuration or if you prefer to modify them. Pay specific attention to the following values:

   - *-DDB2Z_JDBC* should point to Db2 for z/OS JDBC drivers.
   - *-Dblz.httpPort* and *-Dblz.httpPort* should identify the desired server ports.
   - *-Dblz.profilePrefix* and *-Dblz.Keyring* should match the values you specified when you ran BLZRACFL to set up the security settings.

   server.env
   Verify the JAVA_HOME and LIBPATH settings. LIBPATH should point to the Db2 for z/OS JDBC driver library directory.

   server.xml
   The settings in this file cannot be modified.

Copy server started task procedures to PROCLIB and modify them

About this task

You must copy the JCL to run to the IBM WebSphere Liberty server to a system procedure library such as SYS1.PROCLIB.

Procedure

1. Copy BLZZANGL and BLZZSRV from SBLZSAMP to your started task procedure library.
2. If you modified the names of the started tasks when completing the RACF definitions with BLZRACFL, make sure the names of these started task procedures match the RACF definitions you created.
3. Edit and review each of the started task procedures and modify them to meet your configuration:
   - Change the **INSTDIR** parameter to add any path prefix you specified at installation time to the installation path.
   - Change the **USERDIR** parameter to point to the IBM WebSphere Liberty directory that was created when you ran BLZCPWLP. This value should be your BLZWORK directory suffixed with `/wlpuser`, which is `/u/jazz703/wlpuser` by default.
   - Change the **-Dcom.ibm.ws.zos.core.angelName** setting to match the name of your Angel started task.

Start the server processes

About this task

To start the IBM WebSphere Liberty Angel process, which is required, enter the MVS command **S BLZZANGL**. Similarly, to start the IBM WebSphere Liberty process, which is required, enter the MVS command **S BLZZSRV**.

Note: If you modified the started task names, change these commands to use your started task names.

What to do next

After you start the server, review this topic: Running the setup by using Custom setup in the setup wizard in the IBM Engineering Lifecycle Management product documentation.

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 directories:

- `@pathPrefix@/usr/lpp/jazz/v7.0.3/buildsystem`
- `@pathPrefix@/usr/lpp/jazz/v7.0.3/ispfclient`
- `@pathPrefix@/usr/lpp/jazz/v7.0.3/scmtools`

and in the following libraries:

- SBLZLOAD (load modules)
- SBLZMxxx (ISPF messages)
- SBLZSAMP (samples)
- SBLZAUTH (authorized programs)
- SBLZEXEC (CLISTs/REXX).

For detailed installation instructions, see SMP/E installation process.

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 12. 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/jazz703`
@workPath@	The Jazz Team Server working directory.	BLZWORK	`/u/jazz703`

Note: The configuration and work directories require RACF file access permissions. For details, see “Security on z/OS systems” on page 9.

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 13. Variable names and usages		
Variable Name	**Use**	**Default value**
BLZBFAH	The path prefix of the Build agent directory /buildagent	`/usr/lpp/jazz/v7.0.3/buildagent`
BLZBFAC	The Build agent configuration files directory	`/etc/jazz703/ccm`
BLZJAVA	The location of the Java directory	`/usr/lpp/java/J11.0_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 Legacy ISPF gateway for build, deployment, and promotion support

With IBM Engineering Workflow Management (EWM), 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 Legacy ISPF gateway to support build, deployment, and promotion components that are available with a Developer for IBM Enterprise Platforms client access license.

The Interactive ISPF gateway can be used for Enterprise Extensions build translators **only**. For more information, see "Configuring the Interactive ISPF gateway for Enterprise Extensions build support" on page 69.

The following explanations describe how to use the Legacy ISPF gateway for build, deployment, and promotion.

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 Legacy 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 14. Starting commands for EWM components		
Definition	**Command**	**Started from**
Build translator	ISPF command or executable file	Legacy or Interactive ISPF gateway
	TSO command or executable file	Legacy or Interactive ISPF gateway
Build definition	Pre-build command line	z/OS UNIX System Services command
	Post-build command line	z/OS UNIX System Services command

Table 14. Starting commands for EWM components (continued)		
Definition	**Command**	**Started from**
Package definition	Package pre-command	Legacy ISPF gateway
	Package post-command	Legacy ISPF gateway
Deployment definition	Deploy pre-command	Legacy ISPF gateway
	Deploy post-command	Legacy ISPF gateway
	Rollback pre-command	Legacy ISPF gateway
	Rollback post-command	Legacy ISPF gateway
Promotion definition	Pre-build command line	z/OS UNIX System Services command
	Post-build command line	z/OS UNIX System Services command

Programs or executable files can be run by using the ISPF gateway or a UNIX System Services command. z/OS UNIX System Services 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 z/OS UNIX System Services. For more information, see "Starting the Legacy ISPF gateway from a z/OS UNIX System Services command" on page 66.

The following components are included with the Legacy ISPF gateway:

Legacy 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 Legacy 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 (GC19-3623) at http://publibfp.boulder.ibm.com/epubs/pdf/isp2pc00.pdf.

Legacy ISPF gateway startup script: `startispf.sh`

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

In order to start the ISPF gateway, Engineering Workflow Management provides a shell script that is called from the deployment and promotion Ant tasks. This script is installed into `/usr/lpp/jazz/v7.0.3/buildsystem/buildtoolkit/examples/ispfgateway`. As part of standard customization, this script is copied to the Engineering Workflow Management configuration directory, typically `/etc/jazz703/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 in the Engineering Workflow Management Documentation.

Legacy 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 Engineering Workflow Management 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
```

Note: The ISPF.conf allocations are made only when the call method to run a command is ISPF, and not TSO. This also applies to the **allocjob** that can be run to provide additional user allocations. The provided **allocjob** will not be invoked if the call method is TSO.

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 15. Programs run by the deployment component. This table describes the programs that are run during deployment and what those programs do.

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 Engineering Workflow Management 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
isplllib=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 Legacy 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
isplllib=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 (GC19-3623) at http://publibfp.boulder.ibm.com/epubs/pdf/isp2pc00.pdf.

Running the Legacy ISPF Gateway with multi-threaded dependency builds

If the ISPF gateway is going to be run as part of a build that utilizes multiple processes for compilation activities there may be some issues with enqueues on ISPTLIB. This is an issue that is documented in section Protecting table resources in z/OS V2R4 ISPF Dialog Developer's Guide and Reference. Normally with JCL, to stop these enqueues occurring, a temporary data set is allocated as the first data set in the ISPTLIB allocation so that the enqueue is made on the first data set, and that first data set is unique to the JCL being run.

The same issue can occur when running the ISPF Gateway processes in a multi-threaded build mode, or if concurrent builds are run using build agents running with the same user credentials. In order to get around this issue, it is necessary to reallocate ISPTLIB so that the first data set allocated is a unique temporary data set. To do this the ISPF Gateway provides a mechanism to call your own exec to perform other additional allocations. In the ISPF.conf file discussed previously, there is a commented out directive:

```
*allocjob = HLQ.test.exec(alocsamp)
```

Direct the allocjob to a physical exec data that contains a REXX exec to do the allocations. The following example REXX can be used to allocate a new temporary data set and concatenate it in front of your existing ISPTLIB concatenation. In additional see member ISPZISP2 in the Supplied ISPF samples data set, by default ISP.SISPSAMP, for other examples of using this exec.

To ensure uniqueness in the temporary data set name, you need to use the TEMPDSFORMAT(UNIQUE) system option in the ALLOCxx member in PARMLIB. Otherwise, the generated temporary data set name may not be unique. To check what options are in effect, use the DALLOC,OPTION operator command.

```
        /*REXX*/
        /*********************************************************************/
        /* This is a sample test exec job for the "TSO/ISPF Client Gateway"      */
```

```
         /*  may be invoked by ISPF.conf for additional ISPF dataset             */
         /*  concatentations.                                                    */
         /*  Customize accordingly if required.                                  */
         /*                                                                      */
             parse arg $args
             parse var $args USERID
             say '************************************* '
             say 'Running allocation job'
             say 'USERID = 'USERID
             say '************************************* '

          /* Work out current allocation using LISTA */
             x = outtrap('out.','*','concat')
             "lista status sysnames"
             lc = rc
             x = outtrap('off')

          /* Specify DDNAMES you need to reallocate  */
             DDS = 'ISPTLIB'
             DD# = 0
             DSN. = ''

             I = 2
             Do While I <= OUT.0                                      /* @01c
*/
              If Substr (OUT.I,1,1) = ' ' Then
              Do
                Parse var OUT.I DDN . 12 DISP
                If DDN = ' ' Then
                  DSN.DD# = DSN.DD# "'"DSN"'"
                Else
                Do
                  DD# = Wordpos(DDN,DDS)
                  DSN.DD# = "'"DSN"'"
                End
              End
              Else
                Parse var OUT.I DSN .

              I = I +1

             End

             "FREE FILE(ISPTLIB)"

             Call BPXWDYN ("alloc FI(ISPTLIB) cyl space(1,1) RTDSN(DSN)"||,
                           " blksize(27920) lrecl(80) dsorg(po) recfm(f,b) dir(1) new")
             Do i = 1 to Words(DSN.1)
               Call BPXWDYN("alloc FI(TMP00001) dsn("Word(DSN.1,i)") shr")
               Call BPXWDYN("concat ddlist(ISPTLIB,TMP00001)")
             End

             Say 'ISPTLIB : Reallocated with following concatenation -'
             LISTA ST SY HI

          Exit
```

In addition, when running in multi-threaded mode, several temporary ISPF data sets need to be created. The unique identifier that the ISPF Gateway generates is not unique enough when running in multi-threaded mode. However, some changes can be made to create a unique prefix that ISPF can use.
In ISPZXENV, instead of setting the *CGI_ISPPREF* environment variable to &SYSPREF..ISPF.VCMISPF, a prefix can be generated. It is based on the PID (Process Identifier) of the ISPF Gateway process that is running. Add the following code to replace the setting of *CGI_ISPPREF*.

Note: If you do not already have a tailored copy of ISPZXENV, see ISPF gateway environment file: ISPZXENV for instructions on how to create one.

```
/* Running multithreaded we get issues with ISPF.VCMISPF not being unique */
/* Use a hex representation of the PID to uniquelly create ISPF.VCMISPF   */

address syscall 'getpid' ; pid = retval

Say 'Pid:'pid

If Length(Pid) > 7 Then
Do
  CGI_ISPPREF = '&SYSPREF..P$'D2X(Substr(pid,1,5))||,
```

```
                          '.P$'D2X(Substr(pid,6))
End
Else
  CGI_ISPPREF = '&SYSPREF..P'D2X(pid)
```

Starting the Legacy ISPF gateway from a z/OS UNIX System Services 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 Legacy ISPF gateway from the z/OS UNIX System Services command. The BLZGTWY member in *hlq*.SBLZSAMP is a sample executable file for starting the ISPF gateway. You can run this executable file from z/OS UNIX System Services 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:

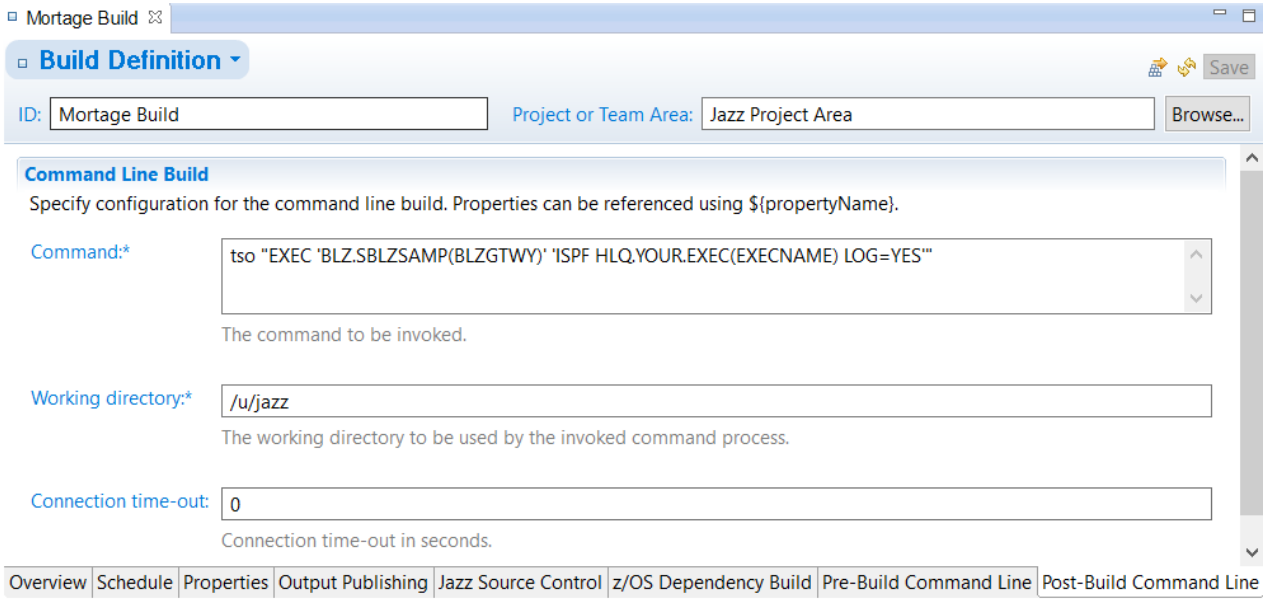
Method
Invocation method, either TSO or ISPF. The TSO method only allocates SYSPROC and SYSEXEC from the ISPF`.conf` file. The ISPF method allocates all the ISPF concatenation from the ISPF`.conf` file.

Exec
Name of the executable file to run.

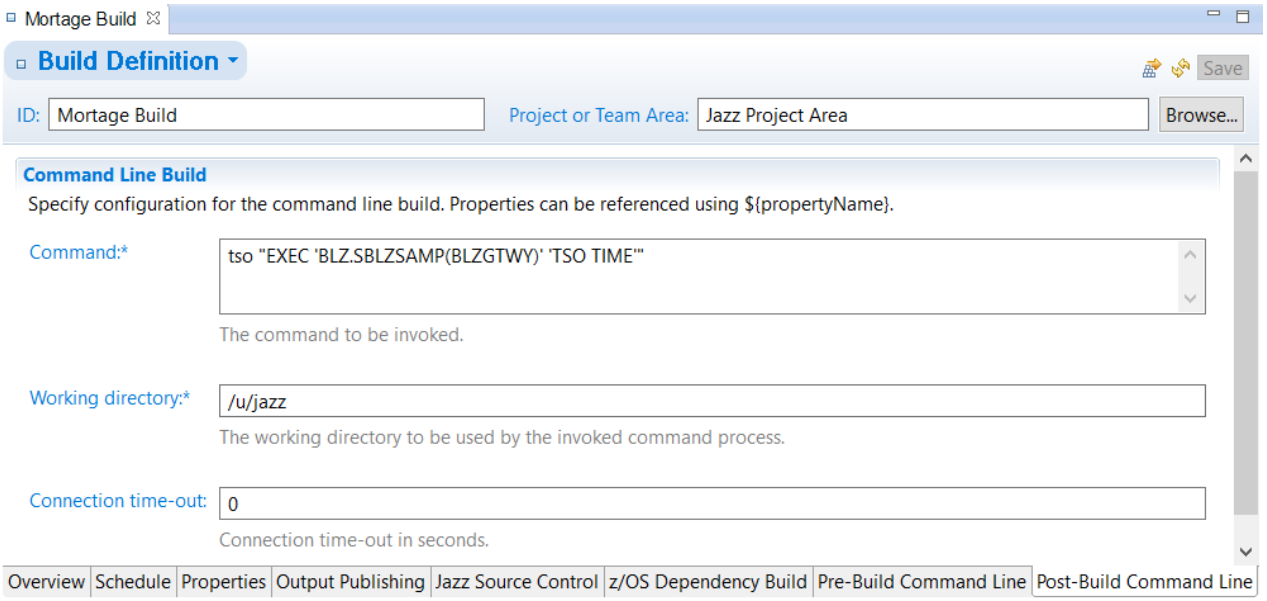
Log
Optional. Specifying LOG=YES writes the full ISPF gateway log, which is useful for debugging problems.

Starting the Legacy 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:



The following example shows how to start the TSO time function with a z/OS UNIX System Services command:



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

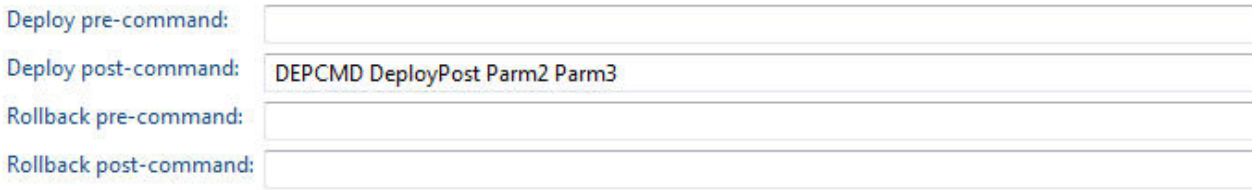
The running of executable files from deployment differs from build and promotion in that it starts the ISPF gateway directly by using the `startispf.sh` shell command. The executable files passed through the pre- and post-deployment commands must be in a certain format. This is true for all of the deployment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



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 Legacy 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="&quot;EXEC 'MYHLQ.SBLZSAMP(BLZGTWY)' 'TSO EX MYHLQ.EXEC(ZIPEXEC) parm1
parm2 parm3' &quot;"/>
       </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 Legacy 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 Legacy ISPF gateway configuration

Verify that the ISPF gateway is properly configured. Any user IDs that run Engineering Workflow Management functions that use the ISPF gateway must have the required permissions that are described in ISPF Planning and Customizing (GC19-3623) at http://publibfp.boulder.ibm.com/epubs/pdf/isp2pc00.pdf.

The Engineering Workflow Management `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/jazz703/ccm/startispf.sh`:

```
export TMPDIR=/yourTemporaryDir
```

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

Configuring the Interactive ISPF gateway for Enterprise Extensions build support

With IBM Engineering Workflow Management (EWM), 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 Legacy ISPF gateway to support build, deployment, and promotion components that are available with a Developer for IBM Enterprise Platforms client access license. For more information, see "Configuring the Legacy ISPF gateway for build, deployment, and promotion support" on page 60.

Alternatively, you can use the Interactive ISPF gateway for Enterprise Extensions build translators. The following explanations describe how to use the Interactive ISPF gateway for build. You cannot use the Interactive ISPF gateway for packaging, deployment, or promotion.

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`.

Prerequisites

To use the Interactive ISPF Gateway for EWM Enterprise Extensions build translators, you need to perform the configuration steps documented by ISPF for this purpose. Specifically, pay attention to the following prerequisites:

- If it is not already installed, install the ISPF PTF that is related to APAR OA63977.
- Read Interactive ISPF Gateway to get an overview of the ISPF Interactive Gateway.
- The Interactive ISPF Gateway is invoked and runs under the address space of the caller. It then uses the z/OS CEA TSO/E address space services to start and communicate with the TSO address space on behalf of the caller. Therefore, the prerequisites described in **System prerequisites for the CEA TSO/E address space services** in **z/OS MVS Programming: Callable Services for High-Level Languages** must be completed. Read Preparing to use the Interactive ISPF Gateway to get a description of the steps needed to configure the Interactive Gateway as well as CEA TSO/E address space services.

Configuring an Enterprise Extensions build definition to use the Interactive ISPF Gateway

You configure an Enterprise Extensions build definition to use the Interactive ISPF Gateway by clicking the **Interactive Gateway** tab of the **Build Definition** editor. For more information, see Specifying z/OS dependency build settings.

Additional considerations

- Interactive Gateway sessions are started when the first build translator that uses such a session is run. These sessions stay active during the rest of the build, assuming that they do not remain unused during the build such that any TSO CEA timeout is reached.
- Interactive Gateway sessions are terminated at the end of the build.
- If the Enterprise Extensions build is running with multiple threads, the best results are achieved if that number of parallel Interactive Gateway sessions is allowed by the gateway configuration and TSO CEA.

  Note: TSO translators and ISPF translators use separate sessions.
- If a session times out (or if the session is for some reason not valid), the ISPF return code is -93 and the command is retried in a new session. Negative return codes are used for infrastructure type errors.
- Translators do not run in an interactive fashion. So, while the TSO/ISPF session is considered interactive, commands are ran one at a time rather than in a request / response mode.

- To run Interactive Gateway supported build translators, the EWM SBLZEXEC library must be included in the SYSEXEC DD concatenation for the TSO logon PROCEDURE that will be used to launch the session. Member BLZISPGW is installed to that library as part of the SMP/E installation of the EWM Build Toolkit FMID HRBT703.
- EWM Interactive ISPF Gateway support uses temporary files that are written to the Rational Build agent `Java` temporary directory and can be controlled by the `java.io.tmpdir` jvm option on the build definition.

The following table shows how programs or executable files can be run during build, deployment, and promotion, and the available support for them. As previously mentioned, **only build translators can use the Interactive ISPF Gateway**.

Table 16. Starting commands for EWM components

Definition	Command	Started from
Build translator	ISPF command or executable file	Legacy or Interactive ISPF gateway
	TSO command or executable file	Legacy or Interactive ISPF gateway
Build definition	Pre-build command line	z/OS UNIX System Services command
	Post-build command line	z/OS UNIX System Services command
Package definition	Package pre-command	Legacy ISPF gateway
	Package post-command	Legacy ISPF gateway
Deployment definition	Deploy pre-command	Legacy ISPF gateway
	Deploy post-command	Legacy ISPF gateway
	Rollback pre-command	Legacy ISPF gateway
	Rollback post-command	Legacy ISPF gateway
Promotion definition	Pre-build command line	z/OS UNIX System Services command
	Post-build command line	z/OS UNIX System Services command

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/v7.0.3/buildagent` during the SMP/E installation. You must install FMID HRBA703 as part of the SMP/E package installation of IBM Engineering Workflow Management (EWM) prior to completing these steps.

Refer to "Installing and configuring Jazz Team Server and the Engineering Lifecycle Management applications on z/OS systems" on page 19 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 IBM Engineering Lifecycle Management v7.0.3 program directories for z/OS (https://www.ibm.com/support/pages/node/6479339).

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. A sample member (BLZCPBFA) in *hlq*.SBLZSAMP is used to create and populate these directories with the required files from the SMP/E installed directory. To fully configure the build

agent, you also need to submit and run BLZCPBTK. See "Creating additional Build System Toolkit directories" on page 59.

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

Table 17. 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/jazz703`
@workPath@	The Jazz Team Server working directory.	BLZWORK	`/u/jazz703`

Note: The Jazz Team Server and CCM application directories require RACF file access permissions. For details, see "Security on z/OS systems" on page 9.

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 18. Variable names and usages

Variable Name	Use	Default value
BLZBFAH	The path prefix of the Build agent directory /buildagent	`/usr/lpp/jazz/v7.0.3/buildagent`
BLZBFAC	The Build agent configuration files directory	`/etc/jazz703/ccm`
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 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 IBM Engineering Workflow Management (EWM) Build System Toolkit 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/v7.0.3/buildsystem/buildtoolkit/examples/startbfa` directory, where *@pathPrefix@* is any prefix you specify when you installed the SMP/E package. This script is required to start the Rational Build Agent to specify credentials so that the agent can connect to the Jazz Team Server (JTS), and to provide you with access to libraries you need. This script is copied to the IBM Engineering Workflow Management (EWM) configuration directory, typically `/etc/jazz703/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 19. Sample script variable strings

Variable	Description
@pathPrefix@	Prefix to the directory path to the Jazz directory. **Note:** This is the prefix to the Jazz directory, so your prefix should include any prefix that you specified as part of the SMP/E installation, and then `/usr/lpp`.
@javaPathPrefix@	Prefix to the directory path to the IBM 64-bit SDK for z/OS Java 2 Technology, typically `/usr/lpp`.
@yourUserid@	The Jazz user ID that will be used to interact with JTS during the build.
@yourPasswordFile@	The Jazz password file as defined by the BLZBPASS job. See Running the Jazz Build Engine on z/OS in the ELM Documentation for information about how to run the BLZBPASS job.
@stepLib@	The STEPLIB DD (data set definition) to use in your z/OS build. For example, `BLZ.SBLZLOAD`
@bfaBinPath@	Directory path to the Rational Build Agent executable directory.
@bfaConfPath@	Directory path to the Rational Build Agent configuration file directory.
@zLang@	(Optional) Encoding used in host files. For example, IBM-037.
@timeout@	(Optional) Timeout value in seconds for build step execution. Default if not specified is 300 seconds.
@workPath@	Directory path where metadata is stored. By default it is `/u/jazz703/build/`.
@yourHomeDirectory@	(Optional) Your home directory. Use this only when you have not defined the *HOME* environment variable somewhere else.
_CMDSERV_BASE_HOME	The location of the ISPF-supplied files so the Rational Build Agent can start the Legacy ISPF gateway. See "Configuring the Legacy ISPF gateway for build, deployment, and promotion support" on page 60 for more information about *_CMDSERV_BASE_HOME*.
_TEMPORARY_UNIT	The temporary unit, for example SYSALLDA, to use the DASD unit of your choice for the allocation of temporary data sets. It does not override the unit specified in the data set definitions. It is used for internal work data sets in EWM processes that run on z/OS, for example in the ISPF client.

Table 19. Sample script variable strings (continued)	
Variable	**Description**
VIO_UNIT	The unit required for VIO in the build, if VIO is not supported or has a different name. For example, set VIO_UNIT to SYSALLDA to use DASD for VIO. It can also be specified in the build engine or as a build property in the build definition.

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.

Important: In an environment with multiple TCP/IP stacks, add:

```
export _BPXK_SETIBMOPT_TRANSPORT=tcp_stack_name
```

to the `startbfa.sh` shell script, which uses the z/OS UNIX System Services *_BPXK_SETIBMOPT_TRANSPORT* variable to specify the name of the TCP/IP stack used.

Notes:

- 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 a z/OS 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.
- 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 from Rocket Software at https://www.rocketsoftware.com/download-center. Specify the `–noprofile` option by adding following two lines to your `bfagent.conf` file:

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

- If build agent logging is enabled in `bfagent.conf` through the activity_log property, the Build Agent shell script will, by default, archive the current log when the agent is started. By default, 5 previous logs will be kept, but this can be changed by altering the *NUM_LOGS* variable.

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 76 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.

Start the build agent by one of these options:

- **Option 1:** If you plan to use Enterprise Extensions, dependency builds, packaging builds, or promotion 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 71.

  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 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 IBM Developer for z/OS UNIX shells can cause conflicts because the ISPF gateway will pick up configuration files from the IBM Developer for z/OS `rse.env`. 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 IBM Developer for z/OS.
- **Option 2:** Start the build agent directly. This method does not set up the environment variables required by Enterprise Extensions. Change directories to `@pathPrefix@/usr/lpp/jazz/v7.0.3/buildagent`, and use the -s option:

  ```
  /usr/lpp/jazz/v7.0.3/buildagent/bfagent -s -f /etc/jazz703/ccm/bfagent.conf
  ```
- **Option 3:** Start the build agent through InetD.

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

Configuring the Rational Build Agent on a z/OS system to start by using InetD

You can configure the Rational Build Agent to start automatically when the server is restarted using the Internet daemon (InetD).

About this task

The Build System Toolkit ships with a shell script (`startbfa.sh`) for z/OS that can be customized to start the Rational Build Agent under z/OS UNIX System Services. The script establishes the required system environment and then starts the agent as a stand-alone daemon. Instead of using the script, you can configure the Rational Build Agent to start automatically with the required system environment when the server is restarted. Complete the following steps to add the build agent to your InetD configuration.

Procedure

Update the InetD configuration files

1. Define the Rational Build Agent (bfagent) service on your z/OS UNIX System Services by adding the following line to the `/etc/services` file:

   ```
   bfagent       8888/tcp    #Rational Build Agent
   ```

2. Replace *8888* with the port number defined in the `bfagent.conf` port property. This line maps the service named *bfagent* to the specified port. When InetD receives a service request on the specified port, it will launch the bfagent service as defined in the InetD configuration file.
3. Add the following line to `/etc/inetd.conf` to define the service for InetD:

   ```
   stream tcp nowait <user>   <agentPath>/bfagent bfagent -f <configPath>/bfagent.conf
   ```

   Where:

   a. *<user>* is the RACF user ID under which the service should be run

   b. *<agentPath>* is the full path to the bfagent executable file

   c. *<configPath>* is the full path to the `bfagent.conf` configuration file
4. After the service has been defined, the InetD daemon must be forced to reread the configuration. There are several options for forcing the daemon to reread the configuration, the easiest method is to stop the daemon and restart it using the following command:

   ```
   kill `cat /etc/inetd.pid`
   /usr/sbin/inetd /etc/inetd.conf
   ```

Set up the build environment

5. When Enterprise Extensions builds are running, the bfagent requires that several environment variables be defined. These variables are typically defined by the `startbfa.sh` script and exported prior to starting the bfagent daemon. To run builds when you are running the agent through InetD, you must define the following variables:

   - HOME
   - ZLANG
   - _TIMEOUT
   - _CMDSRV_BASE_HOME
   - SCM_WORK
   - BLD_TOOLKIT
   - JAVA_HOME
   - JAZZ_USER
   - JAZZ_PASSWORD_FILE
   - STEPLIB
   - ANT_HOME

   To find the expected values of these variables and to determine which ones are required for your environment, see the comments in the `startbfa.sh` script, which is located in the following directory: `@pathPrefix@/usr/lpp/jazz/v7.0.3/buildsystem/buildtoolkit/examples/startbfa/startbfa.sh`.

   Note: Some of these variables might not be required for your setup.
6. Define these variables as properties of a build engine in your project area. To define the variables:

   a) Define a Rational Build Agent type build engine. For more information, see Using the Ant with Enterprise Extensions build definition editor in the Engineering Workflow Management Documentation.

   b) In the build definition editor, specify the port you defined in the `/etc/services` file.

   c) On the build engine editor **Overview** tab, in the **Properties** section, define the variables listed previously.

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 115.

The build agent configuration file, `bfagent.conf` is installed into `@pathPrefix@/usr/lpp/jazz/v7.0.3/buildagent` during the SMP/E installation. For the file to be modified, it must be copied to the configuration directory, typically `/etc/jazz703/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/jazz703/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 73.

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 about the `bfagent.conf` file, see “bfagent.conf file for the Rational Build Agent on z/OS systems” on page 115.

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.

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 IBM Engineering Workflow Management (EWM) 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 15 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 HRDV703) and Rational Build Agent (FMID HRBA703) 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 IBM Developer for z/OS 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 HRDV703. 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” on page 78.
2. Create a started task procedure. For details, see “PROCLIB changes” on page 79.
3. Customize the Job Monitor configuration files. For details, see “Job Monitor configuration file BLZJCNFG” on page 79.
4. Update security definitions. For details, see “Job Monitor security” on page 87.

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 about the PARMLIB definitions listed here. Refer to *MVS System Commands (SA22-7627)* for more information about 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 that follows, 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 Job Monitor actions.

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

Customize the sample Job Monitor configuration member *hlq*.SBLZSAMP(BLZJCNFG), like 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
#HOST_CODEPAGE=IBM-1047
#LIMIT_COMMANDS=USERID
#LIMIT_CONSOLE=LIMITED
#LIMIT_VIEW=NOLIMIT
#LISTEN_QUEUE_LENGTH=5
#LOOPBACK_ONLY=ON
#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 in the following list:

_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 that follow.

- 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 20. LIMIT_COMMANDS command permission matrix

LIMIT_COMMANDS	Job owner is the user	Job owner is not the user
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_CONSOLE
Defines authority level of the EMCS console that is used for executing actions.

Note: The only valid settings are LIMITED (default value) 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.

LOOPBACK_ONLY
Defines whether JES Job Monitor can be accessed from outside this z/OS system.

Note: The only valid settings are ON(default value) and OFF.

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 about 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 z/OS UNIX System Services 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 IBM Developer for z/OS 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=USER,CLASS=A,MSGCLASS=X,REGION=0M
//*
//* COBOL COMPILATION
//*
//COBOL    EXEC PGM=IGYCRCTL,PARM='NODECK,OBJECT,LIB'
//STEPLIB  DD DSN=IGY.V6R2M0.SIGYCOMP,DISP=SHR
//SYSIN    DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO.
       PROCEDURE DIVISION.
       MAIN.
           DISPLAY 'HELLO, RTC-EE.'.
           STOP RUN.
/*
//SYSLIN   DD DSN=USER.V62.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))
//SYSUT8   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT9   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT10  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT11  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT12  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT13  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT14  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT15  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSMDECK 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=USER.V62.OBJ,DISP=SHR
//         DD DSN=CEE.SCEELKED,DISP=SHR
//SYSLMOD  DD DSN=USER.V62.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:

i) 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)>
```

ii) Set the working directory to a fully qualified z/OS UNIX System Services 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 84 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.

i) Click **Add**.

ii) Select String as the property type and click **OK**.

iii) Specify HLQ as the name.

iv) Enter the HLQ of the target data sets and click **OK**.

i. Specify the following values on the Build Command Line tab:

i) 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=USER,CLASS=A,MSGCLASS=X,REGION=0M
//*
//* COBOL COMPILATION
//*
//COBOL    EXEC PGM=IGYCRCTL,PARM='NODECK,OBJECT,LIB'
//STEPLIB  DD DSN=IGY.V6R2M0.SIGYCOMP,DISP=SHR
//SYSIN    DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO.
       PROCEDURE DIVISION.
       MAIN.
```

```
           DISPLAY 'HELLO, RTC-EE.'.
           STOP RUN.
/*
//SYSLIN   DD DSN=USER.V62.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))
//SYSUT8   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT9   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT10  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT11  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT12  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT13  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT14  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSUT15  DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSMDECK 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=USER.V62.OBJ,DISP=SHR
//         DD DSN=CEE.SCEELKED,DISP=SHR
//SYSLMOD  DD DSN=USER.V62.LOAD(HELLO),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*
```

ii) Set the working directory to a fully qualified z/OS UNIX System Services 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:

i) In the Team Artifacts view, select the build definition, right-click, and select **Request Build**.

ii) Click **Submit**.

iii) If a dialog is displayed that states that the build engine does not appear to be processing requests, click **OK** to submit the request.

iv) 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 21. Job Monitor console commands.

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 21 on page 88 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 79.

Table 22. Job Monitor command permission matrix.

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 23. Extended JESSPOOL profiles.

Extended JESSPOOL profiles

Header	**JESSPOOL profile**	**Required access**
Hold	nodeid.userid.jobname.jobid	ALTER

Table 23. Extended JESSPOOL profiles.

Extended JESSPOOL profiles

(continued)

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

Use the following substitutions in the preceding table:

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 79. 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 79.

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. If your security is set up as documented in this product documentation, JES commands that you issue from this console will fail the security check, unless you are authorized to issue 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 about 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 79.

Table 24. Job Monitor browse permission matrix.

Job Monitor browse permissions

	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 about 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 87.

- `LIMIT_VIEW={USERID | NOLIMIT}`

  Define which spool files can be browsed. For more information, see "Access to spool files" on page 90.

Note: Details on the BLZJCNFG directives are available in "Job Monitor configuration file BLZJCNFG" on page 79.

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" on page 91
- "Define JES command security" on page 91

Refer to the *RACF Command Language Reference (SA22-7687)*, for more information about 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('EWM')
  ```

- ```
  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 79.

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('EWM'))
  ```
  ```

- `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 25 on page 92 and Table 26 on page 92 show the operator commands issued for JES2 and JES3, and the discrete security profiles that you can use to protect them.

Table 25. 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 26. JES3 Job Monitor operator commands

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

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 87 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 90 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 product documentation, 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 “Using the Jazz Build Engine on z/OS” on page 93.
2. If you are installing the IBM Developer for z Systems® integration feature, see “Integrating with IBM Developer for z/OS” on page 95
3. 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” on page 78.

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 in the IBM Engineering Workflow Management Documentation.

Before performing the setup steps, review the Manage builds with the IBM Engineering Workflow Management Build lesson in the *Get started with Engineering Workflow Management* tutorial of the Engineering Workflow Management Documentation.

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 Engineering Workflow Management 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 about creating an encrypted password file, see Creating a password file using the sample BLZBPASS job.

   Note: (Optional) You can now use application password with OpenID Connect. For more information, see Obtaining an application password.
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 Engineering Workflow Management 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.

Integrating with IBM Developer for z/OS

There are additional installation and configuration steps required to integrate IBM Developer for z/OS and IBM Engineering Workflow Management (EWM).

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 IBM Developer for z/OS Host Configuration Guide in the IBM Developer for z/OS documentation.

These steps require that you have already installed the Build System Toolkit for z Systems.

About this task

1. Locate the IBM Explorer for z/OS configuration directory. The default location is `/etc/zexpl`. See the IBM Explorer for z/OS documentation for details.
2. Create a file named `rse-ewm.env` in the IBM Explorer for z/OS configuration directory using the contents of sample member BLZZEXPL from *hlq*.SBLZSAMP, where *hlq* is the high-level qualifier specified during the SMP/E installation. The BLZZEXPL member contains the IBM Engineering Workflow Management (EWM) CLASSPATH assignment. The sample member also contains stubbed declarations of both the *LIB_PATH* and *SCM_WORK* environment variables. For clarification about how to set these environment variables, review the comments in the new `rse-ewm.env` file. Ensure that the file permissions for the new `rse-ewm.env` are the same as existing environment variable files.
3. Restart the IBM Explorer for z/OS RSE daemon. The default name for this started task is RSED.

Installing the integrated client

Installing the IBM Engineering Workflow Management (EWM) Integration feature requires that you install it after IBM Developer for z/OS is installed, or with IBM Developer for z/OS (if IBM Engineering Workflow Management is already installed.)

Before you begin

These instructions assume that you are installing the Engineering Workflow Management client before you install the IBM Developer for z/OS client. If you already have IBM Developer for z/OS client installed and you want to add the Engineering Workflow Management client, see “Installing Engineering Workflow Management on IBM Developer for z/OS” on page 96.

Notes:

- To know the required IBM Developer for z/OS version, see Hardware and software requirementsin the Engineering Workflow Management Documentation.
- Not all Eclipse-based products work with all versions of the Eclipse integrated development environment (IDE). If you are installing Engineering Workflow Management or IBM Developer for z/OS into an existing Eclipse IDE, check the system requirements for both products to make sure that they are compatible.

Procedure

1. Install the Engineering Workflow Management client using the Engineering Workflow Management launchpad.
2. Install the IBM Developer for z/OS client using the IBM Developer for z/OS launchpad.

   Note: Make sure that you select the same IBM Installation Manager package that you used in your Engineering Workflow Management client installation.

3. Select **Engineering Workflow Management Integration feature** in the Installation Manager.

   Important: This integration feature is only available for installation if Engineering Workflow Management is already installed into the same Installation Manager package into which you are installing IBM Developer for z/OS.
4. Follow the remaining Installation Manager wizard instructions to complete the installation.

Installing Engineering Workflow Management on IBM Developer for z/OS

If you already have the IBM Developer for z/OS client installed, you can install the IBM Engineering Workflow Management (EWM) 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 Engineering Workflow Management client, you cannot install the Engineering Workflow Management Integration feature. Perform the following steps to install Engineering Workflow Management and the extension that is the Engineering Workflow Management integration feature.

Procedure

1. Install the Engineering Workflow Management client using the Engineering Workflow Management launchpad.
2. Select the option to install into the same package group as the IBM Developer for z/OS client.

   Note: Not all Eclipse-based products work with all versions of the Eclipse integrated development environment (IDE). If you are installing Engineering Workflow Management client or IBM Developer for z/OS 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 Engineering Workflow Management Integration feature.
5. Follow the remaining Installation Manager wizard instructions to finish the installation.

Installing the IBM Engineering Workflow Management 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 IBM Engineering Workflow Management (EWM) repository.

The Engineering Workflow Management Interactive System Productivity Facility (ISPF) client provides an ISPF interface to perform various Engineering Workflow Management functions. You can use the interface to edit, check-in, deliver, and build code that is stored in a Engineering Workflow Management repository.

The ISPF client is installed as part of the Build System Toolkit during the SMP/E installation of FMID HRBT703. 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.

ISPF client security

The ISPF client must be secure so that only authorized users can access stored files.

The IBM Engineering Workflow Management (EWM) ISPF client provides mainframe access to source code that is stored in an EWM 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 EWM 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 EWM. 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 about 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 EWM 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" on page 97
- "OMVS segment for ISPF client users" on page 98
- "Data set profiles for the ISPF client" on page 98
- "ISPF daemon started task on System z" on page 99
- "ISPF daemon as a secure z/OS UNIX System Services server" on page 100
- "Application protection for the ISPF daemon" on page 100
- "ISPF PassTicket support " on page 101
- "z/OS UNIX System Services program-controlled files for the ISPF daemon" on page 102

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

Security settings and classes for the ISPF client

Activate security settings and classes for the ISPF client with RACF commands.

IBM Engineering Workflow Management (EWM) 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 EWM 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)
```

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 theIBM Engineering Workflow Management (EWM) 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))
```

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.

Data set profiles for the ISPF client

Provide access to IBM Engineering Workflow Management (EWM) data sets for users.

READ access for users and ALTER for system programmers is sufficient for most EWM 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('EWM - HLQ STUB')
```

-
```
ADDSD ‘BLZ.**' UACC(READ)
DATA('EWM')
```

-
```
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('EWM - HLQ STUB')
  OWNER(IBMUSER) SUPGROUP(SYS1)
```

- ```
  ADDSD BLZ.**' UACC(NONE)
    DATA('EWM')
  ```
- ```
  ADDSD 'BLZ.SBLZLOAD' UACC(NONE)
    DATA('EWM')
  ```
- ```
  ADDSD 'BLZ.SBLZEXEC' UACC(NONE)
    DATA('EWM')
  ```
- ```
  ADDSD 'BLZ.SBLZMENU' UACC(NONE)
    DATA('EWM')
  ```
- ```
  ADDSD 'BLZ.SBLZMENU' UACC(NONE)
    DATA('EWM')
  ```
- ```
  ADDSD 'BLZ.SBLZSAMP' UACC(NONE)
    DATA('EWM')
  ```

- 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 EWM 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 previous sample.

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('EWM - ISPF DAEMON')
  OMVS(UID(#user-id-ispf) HOME(/tmp) PROGRAM(/bin/sh) ASSIZEMAX(2147483647))
  DATA('EWM')
  ```
- ```
  RDEFINE STARTED BLZISPFD.* DATA('EWM - ISPF DAEMON - START')
  STDATA(USER(STCISPF) GROUP(STCGROUP) TRUSTED(NO))
  ```
- ```
  RDEFINE STARTED BLZISPFS.* DATA('EWM - 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).

ISPF daemon as a secure z/OS UNIX System Services 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`

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('EWM')
  ```
- `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 ” on page 101, 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.

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 following sample RACF commands.

- ```
  RDEFINE PTKTDATA BLZAPPL UACC(NONE) SSIGNON(KEYMASKED(key16))
  APPLDATA('NO REPLAY PROTECTION – DO NOT CHANGE')
  DATA('EWM')
  ```

  The following example shows the command with the key16 value replaced:

  ```
  RDEFINE PTKTDATA BLZAPPL UACC(NONE) -
  DATA('EWM') -
  APPLDATA('NO REPLAY PROTECTION - DO NOT CHANGE') -
  SSIGNON(KEYMASKED(0123456789ABCDEF))
  ```

- ```
  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 previously.
2. 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.
3. 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 100. 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 104 for details.

A profile in the FACILITY class is provided to define who can generate PassTickets. This profile must exist or PassTicket generation fails. The user ID requesting a PassTicket must have read access to this profile. Use the following sample RACF commands to create a profile:

- ```
  RDEFINE FACILITY BLZ.CONNECT.BLZAPPL UACC(NONE) -
  DATA('EWM') –
  ```

- ```
  PERMIT BLZ.CONNECT.BLZAPPL CLASS(FACILITY) ACCESS(READ) ID(JAZZWORK)
  ```

- ```
  SETROPTS RACLIST(FACILITY) REFRESH
  ```

Notes:

1. This profile uses the application ID as the last part of the profile. It must match the application ID being used to generate the PassTicket. If the application ID is set to a different value, this profile must be set up using the changed application ID.
2. Following normal System Authorization Facility (SAF) rules, a generic profile can cover all application IDs on a system. For example : `BLZ.CONNECT.*`.
3. All users of the ISPF daemon need to have READ access to this profile. You can provide access by adding a PERMIT for the specific user, a PERMIT for a group that the user is in, or by specifying UACC=READ on the profile so that all users can generate PassTickets.

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.

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

z/OS UNIX System Services 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 System Services 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 IBM Engineering Workflow Management (EWM) programs.

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

Notes:

1. The setup might be different if you use a security product other than RACF. Consult the documentation of your security product for more information.
2. The SMP/E installation of IBM Engineering Workflow Management sets the program-control bit for internal programs, when it is available.
3. Use the **ls -Eog** z/OS Linux 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 a-s- 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/jazznnn/ccm (where nnn is the version).

Note: The daemon on z/OS needs to be restarted after a server rename. The ISPF daemon should disconnect itself after the server rename.

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 following instructions, 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 that follows, you must provide:

1. The ISPF daemon port. The default port is 4152.
2. The home directory where Engineering Workflow Management (EWM) is installed. The default directory is `/usr/lpp/jazz/v7.0.3`.
3. The location of the configuration files. The default location is `/etc/jazz703/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/v7.0.3',
//             CNFG='/etc/jazz703/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 106 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/jazz703'
  //*
  //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/v7.0.3/ispfclient/bin/ispfdmn.sh
                                                        4197 /etc/jazz/ccm
  ```

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 in `/etc/jazz703/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 59.

You can edit the ispfdmn.conf file with ISPF option 3.17. The ispfdmn.conf file must be customized to match your system environment. When you are using a US code page, comment lines start with a pound sign (#). Data lines can have a directive and its assigned value only, 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 with the _ISPF_CLEANUP_THRESHHOLD environment variable. The cleanup interval logs out any sessions that are inactive for the specified time within the specified threshold time. For example, with _ISPF_CLEANUP_INTERVAL set to 900000 and

_ISPF_CLEANUP_THRESHHOLD set to 86400000, the daemon scans every 15 minutes and discards any sessions inactive for 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 path is in a hierarchical file system (HFS) location where the daemon writes its location information. For example, you could set this path to `/u/work/ccm`.

JAVA_HOME
Specifies the path to the Java home directory. The default is `/usr/lpp/java/J11.0_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.

RTC_HOME
Specifies the path to the IBM Engineering Workflow Management (EWM) home directory. The default is `/usr/lpp/jazz/v7.0.3/`. Change to match your EWM 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 path to your work directory, which is `/u/jazz703/ccm` by default. Developers running the EWM 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:

_BPXK_SETIBMOPT_TRANSPORT
Specifies the TCP/IP stack name if the site is using multiple TCP/IP stacks and the default TCPIP stack is not being used.

_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 that is specified or allowed to default, subsequent log entries are logged in a new log file.

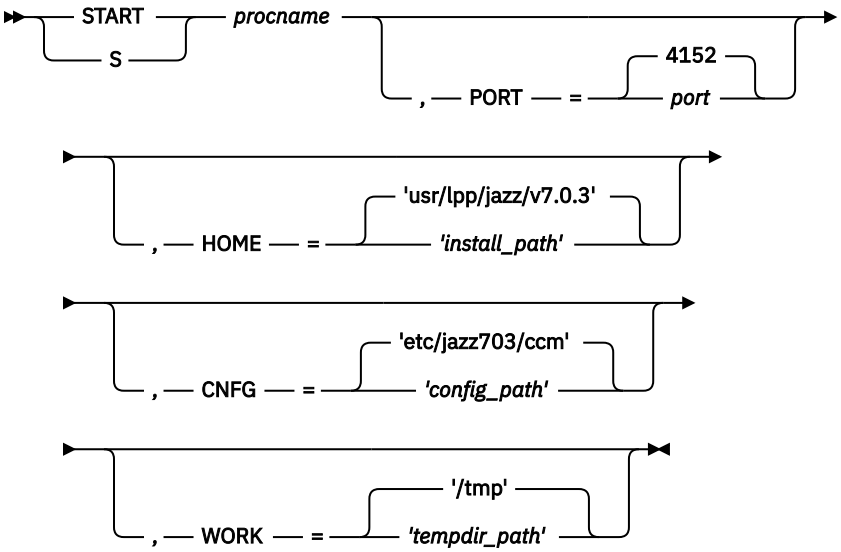
_ISPF_DAEMON_LOG_CONF
Specifies the absolute path to where the ISPF daemon trace log files are stored, for example, `/u/username/conf/ccm/commons-logging.properties`.

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**.

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.



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/v7.0.3` used to install IBM Engineering Workflow Management (EWM). The default is `/usr/lpp/jazz/v7.0.3`.

Note: The z/OS UNIX path is case sensitive and it must be enclosed in single quotes (') to preserve lowercase characters.

CNFG
Absolute location of the configuration files stored in z/OS UNIX. The default is '/etc/jazz703/ccm'.

Note: The z/OS UNIX path is case sensitive and it must be enclosed in single quotes (') to preserve lowercase 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 lowercase 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 beginning with the >> symbol and following the horizontal line (the main path) until you reach the >< symbol.

The following symbols are used in syntax diagrams:

Symbol
Description

>>
Marks the beginning of the syntax diagram.

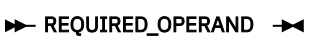
>
Indicates that the syntax diagram is continued.

|
Marks the beginning and end of a fragment or part of the syntax diagram.

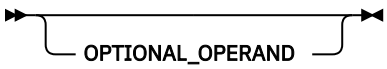
><
The end of the syntax diagram.

The following types of operands are used in syntax diagrams:

- Required operands are displayed on the main path line:



- Optional operands are displayed underneath the main path line:

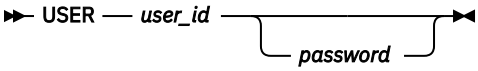


- Default operands are displayed over the main path line:

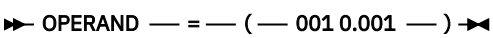
Operands are classified as keywords or variables:

- 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.

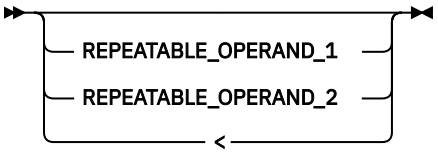
In the following example, the USER command is a keyword. The required variable parameter is user_id, and the optional variable parameter is password. Replace the variable parameters with your own values:



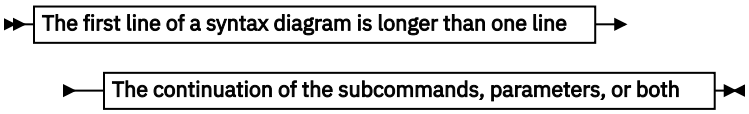
If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, equal signs, and blank spaces), you must code the character as part of the syntax. In this example, you must code OPERAND=(001 0.001):



An arrow returning to an earlier part of the syntax main line in a group of operands means that more than one can be selected, or that a single one can be repeated:



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:



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 following the main diagram:

Syntax fragment

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 96 are required to use the ISPF client and daemon.

The IBM Engineering Workflow Management (EWM) 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.

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 EWM 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

Installation verification process (IVP) for z/OS

The IBM Engineering Lifecycle Management (ELM) installation verification process (IVP) verifies information about settings that you configured during installation on z/OS.

The installation verification process (IVP) provides information about the settings of the following installation configurations:

- General installation
- IBM Engineering Workflow Management (EWM) ISPF client
- Rational Build Agent
- Legacy ISPF gateway.

The IVP checks for things that might cause problems with the listed settings, if they are not set up correctly, such as:

- RACF permissions on data sets
- RACF on general resource profiles
- UNIX Systems Services permissions on files and directories
- Space restrictions on configuration and work directories
- Legacy ISPF gateway set up
- APF authorizations
- Necessary PARMLIB members are correctly updated.

The IVP has the following restrictions:

- The user ID that runs the IVP process must have sufficient z/OS UNIX authority.
- The IVP process uses SDSF REXX calls and RACF REXX calls. If the user ID that runs this process does not have System Display and Search Facility (SDSF) or RACF authority, the commands fail and return a message indicating the lack of authority.
- If SDSF or RACF is not installed on your system, the commands fail and return a message indicating that the programs are not installed.

To run the IVP, sample member BLZIVP is provided in *hlq*.SBLZSAMP. Configure member BLZIVP using the instructions contained in the member. The variables mentioned in the sample job are listed in the following table with an explanation of their use and default value:

Table 27. BLZIVP variable names and usage

Variable name	Usage	Default value
PREFIX	The data set prefix specified during the SMP/E installation.	BLZ
HOME	The path prefix specified during the SMP/E installation.	`/usr/lpp/jazz/v7.0.3`
CONF	The Jazz Team Server configuration files directory.	`/etc/jazz703`
WORK	The Jazz Team Server working directory.	`/u/jazz703`

Table 27. BLZIVP variable names and usage (continued)		
Variable name	**Usage**	**Default value**
ICONV	The location of the iconv installation directory.	`/bin/iconv`
BLZDSAUT	The SBLZAUTH data set containing the BLZPASTK member	BLZ.SBLZAUTH
GENIVP	Whether to run the general IVP	Y
ISPFIVP	Whether to run the EWM ISPF client IVP	Y
BFAIVP	Whether to run the Rational Build Agent IVP	Y
GTWYIVP	Whether to run the ISPF gateway IVP	Y
JMONIVP	Whether to run Job Monitor IVP	Y
APPSIVP	Whether to run ELM server applications IVP	N
SRVRIVP	Whether to run application server IVP	N
BLZISPFD	Started task name for the EWM daemon start task	BLZISPFD
BLZISPFS	Started task name for the EWM daemon stop task	BLZISPFS
BLZBFA	Started task name for the Rational Build Agent	BLZBFA
BLZJTASK	Started task name for the Job Monitor	BLZJMON
BLZJPORT	Port for the Job Monitor	6716
CONFGID	Group id for JAZZCONF for configuration directories	1
WORKGID	Group id for JAZZWORK for work directories	2
DB2SYS	Db2 subsystem for server repositories	DSNA
DB2RUN	RUNLIB for Db2 subsystem	DSNA10
RUNLIB.LOAD DSNTEP2	DSNTEP procedure name	DSNTEP10
BLZLIBAN	Started task for IBM WebSphere Liberty Angel process	BLZZANGL
BLZLIBST	Started task for IBM WebSphere Liberty server	BLZZSRV
SUPPRESS	Suppress information messages	N

Use the SUBMIT command to submit the modified JCL and check the job log. The output from the installation verification process is in DD RTCIVP.

The following table shows the return codes for the IVP.

Table 28. BLZIVP return codes	
Return code	**Explanation**
0	Indicates that the installation verification has run successfully without any major issues.
4	Indicates that the installation verification process has run, but with warnings.
8	Indicates that the installation verification process has run, but with errors.

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. For information about components that are installed on other systems, review Deployment and installation considerations in the ELM Documentation.
2. To install a client, see Installing an Engineering Workflow Management client in the ELM Documentation.

Chapter 5. Appendix

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 in the installed `bfagent.conf` file.

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 `startbfa.sh` script contains settings to control how many logs are saved when the agent is restarted. 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`**.

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.

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.

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: TLSV1_2 or TLSV1_3.

gsk_ssl_v3_cipher_specs_expanded
When TLSV1_3 is used, this setting provides the 4-digit cipher specification. The default is **1301**.

gsk_ssl_server_tls_key_shares
When TLSV1_3 is used, this setting provides the SSL server TLS key shares. The default is **002300250029**.

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**.

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.

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 TLSv1.2, TLSv1.3. The protocol must match the protocol used by the server. When enabled, TLSv1.2, or TLSv1.3, 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.

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.

Notices for IBM Engineering Lifecycle Management

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

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

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

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

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

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

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

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

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

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation

North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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

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

Trademarks

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

Terms and conditions for product documentation

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

Applicability

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

Personal use

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

Commercial use

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

Rights

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

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

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

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

IBM Online Privacy Statement

Privacy policy considerations

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

Depending upon the configurations deployed, the Software Offering may use session and persistent cookies that collect each user’s user name and password for purposes of session management, authentication, enhanced user usability, and single sign-on configuration. These cookies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at https://www.ibm.com/privacy/details/us/en/ in the section entitled “Cookies, Web Beacons and Other Technologies”.