# IBM Developer for z/OS
# Version 15.0

# *Host Configuration Guide*

SC27-9933-03

**Note**

Before using this information, be sure to read the general information under “Notices” on page 93.

**Fourth edition (May 2022)**

This edition applies to IBM® Developer for z/OS® Version 15.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send your comments by mail to the following address:

IBM Corporation
Attn: Information Development Department 53NA
Building 501 P.O. Box 12195
Research Triangle Park NC 27709-2195
USA

You can fax your comments to: 1-800-227-5088 (US and Canada)

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

**© Copyright International Business Machines Corporation 2000, 2024.**
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

# Figures

# Tables

# About this document

You can use Host Configuration Assistant for Z Development (HCA) to generate customized checklists for installing and configuring IBM Developer for z/OS, IBM Explorer for z/OS, and their components and companion products, such as z/OS Source Code Analysis. You can also find the configuration information for other host components such as z/OS Explorer, z/OS Debugger, and Dependency Based Build. After generating a configuration checklist, refer to this guide and the other host configuration guides for instructions.

This document discusses the configuration of IBM Developer for z/OS on your z/OS host system.

The following names are used in this document:

- *IBM Explorer for z/OS* is called *z/OS Explorer*.
- *IBM z/OS Debugger* is called *z/OS Debugger*.
- *IBM Developer for z/OS* is called *Developer for z/OS*.
- *IBM z/OS Explorer Extensions* is called *z/OS Explorer Extensions*.
- *IBM z/OS Source Code Analysis* is called *z/OS Source Code Analysis*.
- *IBM Developer for z/OS Interface for CA Endevor® SCM* is called *CA Endevor® SCM RAM*.
- *Common Access Repository Manager* is abbreviated to *CARMA*.
- *IBM z/OS Automated Unit Testing Framework* is called *ZUnit*.
- *z/OS UNIX System Services* is called *z/OS UNIX*.
- *Customer Information Control System Transaction Server* is called *CICSTS*, abbreviated to *CICS®*.

The host components of Developer for z/OS are shared between multiple products, and therefore have a product-neutral name. This publication discusses configuration of the following FMIDs:

- HHOPxxx, IBM z/OS Explorer Extensions

  This FMID adds additional services to z/OS Explorer that can be used by the Developer for z/OS client.
- HAKGxxx, IBM z/OS Source Code Analysis

  This FMID provides a batch interface for source code analysis.

This document is part of a set of documents that describe Developer for z/OS host system configuration. Each of these documents has a specific target audience. To complete the Developer for z/OS configuration, you are not required to read all documents.

- *IBM Developer for z/OS Host Configuration Guide* (this document) describes in detail all planning tasks, configuration tasks, and options (including optional ones) and provides alternative scenarios.
- *IBM Developer for z/OS Host Configuration Reference* describes Developer for z/OS design and gives background information for various configuration tasks of Developer for z/OS, z/OS components, and other products (such as WLM) related to Developer for z/OS.

For the most up-to-date versions of this document, see the Developer for z/OS documentation available at https://www.ibm.com/docs/en/developer-for-zos/latest .

For the most up-to-date versions of the complete documentation, including installation instructions, white papers, podcasts, and tutorials, see the library page of the IBM Developer for z/OS website (http://www.ibm.com/support/docview.wss?uid=swg27048563).

# Who should use this document

This document is intended for system programmers who are installing and configuring IBM Developer for z/OS.

© Copyright IBM Corp. 2000, 2024

This document lists in detail the steps that are needed to do a full setup of the product, including some non-default scenarios. Background information that can help you to plan and execute the configuration can be found in the *IBM Developer for z/OS Host Configuration Reference*. To use this document, you must be familiar with the z/OS UNIX System Services and MVS™ host systems.

# Part 1. Host Configuration Guide

© Copyright IBM Corp. 2000, 2024

# Chapter 1. Planning

You can use Host Configuration Assistant for Z Development (HCA) to generate customized checklists for installing and configuring IBM Developer for z/OS, IBM Explorer for z/OS, and their components and companion products, such as z/OS Source Code Analysis. You can also find the configuration information for other host components such as z/OS Explorer, z/OS Debugger, and Dependency Based Build. After generating a configuration checklist, refer to this guide and the other host configuration guides for instructions.

Use the information in this chapter to plan the installation and deployment of Developer for z/OS. The following subjects are described:



For a complete list of the Developer for z/OS hardware and software requirements, including prerequisites and corequisites, see the Software Product Compatibility Reports (SPCR) tool (https://www.ibm.com/software/reports/compatibility/clarity/index.html).

## Migration considerations

Chapter 8, "Migration guide," on page 75 describes changes to the installation and configuration of this release with respect to previous releases of the product. Use this information to plan your migration to the current release of Developer for z/OS.

## Planning considerations

### Product overview

Developer for z/OS consists of a client, installed on the user's personal computer, and a server, installed on one or more host systems. Both client and host are installed on top of IBM Explorer for z/OS. This documentation contains information for a z/OS host system.

The client provides developers with an Eclipse-based development environment that facilitates a uniform graphical interface to the host, and that, among other things, can offload work from the host to the client, saving resources on the host.

The host portion consists of several permanently active tasks and tasks that are started ad hoc. These tasks allow the client to work with the various components of your z/OS host system, such as MVS data sets, TSO commands, z/OS UNIX files and commands, job submit, and job output. The host components of Developer for z/OS are shared by multiple products offered by IBM.

Developer for z/OS enhances the basic access functionality provided by z/OS Explorer. Developer for z/OS can, for example, interact with subsystems and other application software on the host system, such as CICS, and Software Configuration Managers (SCMs), if Developer for z/OS is configured to do so, and if these co-requisite products are available.

For information about Developer for z/OS itself, how it interacts with your system, and with the prerequisite and co-requisite products, see the *Developer for z/OS Host Configuration Reference*. The *IBM Explorer for z/OS Host Configuration Reference* gives similar information for z/OS Explorer, which is a requisite for Developer for z/OS.

© Copyright IBM Corp. 2000, 2024

To learn more about the functionality that is offered by Developer for z/OS, see the Developer for z/OS IBM Documentation at https://www.ibm.com/docs/en/developer-for-zos/latest, or your local IBM representative.

## Skill requirements

SMP/E skills are needed for a Developer for z/OS host installation.

The configuration of Developer for z/OS requires more than the typical system programming permissions and expertise, so assistance from others might be needed. Table 3 on page 6 and Table 4 on page 6 list the administrators who are needed for the required and optional customization tasks.

## Time requirements

The amount of time that is required to install and configure the Developer for z/OS host system components depends on various factors such as these:

- The current z/OS UNIX and TCP/IP configuration
- The availability of prerequisite software and maintenance
- The availability of a user, who has successfully installed the client, to test the installation and report any problems that might occur

Experience has shown that the installation and configuration process for the Developer for z/OS host system requires from one to two days to complete. This time requirement does not include the installation and configuration of IBM Explorer for z/OS, which is a required product, nor does it include the installation and configuration of other FMIDs provided together with Developer for z/OS. This time requirement is for a clean installation performed by an experienced system programmer. If problems are encountered, or if the required skills are not available, the setup will take longer.

# Preinstallation considerations

For detailed instructions on the SMP/E installation of the product, see *Program Directory for IBM z/OS Explorer Extensions*.

The IBM Developer for z/OS FMIDs are shared by multiple IBM products, so it is possible you already have installed one or more of the FMIDs. In this case you can reuse the existing installation and go directly to the configuration tasks.

The Developer for z/OS servers are single-system minded, and are not SYSPLEX aware. If you are using the servers in a SYSPLEX, you must ensure that the data requested by the end users (data sets, job output, z/OS UNIX files) is available on the system Developer for z/OS is installed. See “Predeployment considerations” on page 8 for cloning Developer for z/OS to other systems.

To run multiple instances of Developer for z/OS on a single host system, see "Running multiple instances" in the *IBM Explorer for z/OS Host Configuration Reference* .

The file system in which Developer for z/OS is installed must be mounted with the `SETUID` permission bit on (this is the system default). Mounting the file system with the `NOSETUID` parameter prevents Developer for z/OS from creating the user's security environment, and rejects the connection requests of the client. The same is true for the file systems hosting z/OS Explorer, Java™, and z/OS UNIX binaries.

## Installation user ID

The user ID that is used to install Developer for z/OS, or to install maintenance, must have at least the following attributes:

- TSO access (with a normal region size).

  **Note:** A large region size is required for the user ID that runs the Installation Verification Programs (IVPs) because functions requiring a lot of memory (such as Java) are executed. You should set the region size to 131072 kilobytes (128 megabytes) or higher.

- An OMVS segment defined to the security system (for example, RACF®), both for the user ID and its default group.
  - The HOME field must refer to a home directory that is allocated for the user, with READ, WRITE, and EXECUTE access.
  - The PROGRAM field in the OMVS segment should be `/bin/sh` or other valid z/OS UNIX shell, such as `/bin/tcsh`.
  - The user ID's default group requires a `GID`.
- `UID=0` or READ authorization to the `BPX.SUPERUSER` profile in the `FACILITY` class.
- If the `BPX.FILEATTR.APF` or `BPX.FILEATTR.PROGCTL` profiles are defined in the `FACILITY` class, READ access to these profiles.
- READ, WRITE, and EXECUTE access to the `/tmp` directory (or a directory referenced in the `TMPDIR` environment variable).

## Requisite products

Developer for z/OS has a list of prerequisite software that must be installed and operational before the product will work. There is also a list of corequisite software to support specific features of Developer for z/OS. These requisites must be installed and operational at runtime for the corresponding features to work as designed.

For a complete listing of the Developer for z/OS software requirements including prerequisites and co-requisites, see the Software Product Compatibility Reports (SPCR) tool (https://www.ibm.com/software/reports/compatibility/clarity/index.html).

Plan ahead to have these requisite products available, as it might take some time, depending on the policies at your site. The key requisites for a basic setup are:

- z/OS 2.3 or higher
- IBM Explorer for z/OS 3.2
- Latest service release of Java 8.0 or higher (31 or 64 bit)

## Required resources

Developer for z/OS requires the allocation of the systems resources listed in Table 1 on page 5. The resources listed in Table 2 on page 5 are required for optional services. Plan to have these resources available because, depending on the policies at your site, it might take some time to get them.

*Table 1. Required resources*

| Resource | Default value | Information |
|---|---|---|
| MVS build procedures | ELAXF* | "PROCLIB changes" on page 19 |

*Table 2. Optional resources*

| Resource | Default value | Information |
|---|---|---|
| LPA data set | FEL.SFELLPA | "LPA definitions in LPALSTxx" on page 16 |
| port range for host-confined use | any available port is used | • Chapter 3, "Common Access Repository Manager (CARMA)," on page 27<br>• Chapter 4, "Host-based code analysis," on page 57<br>• "xUnit support for CICS applications (ZUnit)" on page 65 |
| CICS JCL update | FEL.SFELLOAD | "CICS bidirectional language support" on page 69 |

| Table 2. Optional resources (continued) | | |
|---|---|---|
| **Resource** | **Default value** | **Information** |
| PROCLIB update | • AZU*<br>• AKG* | • “xUnit support for Enterprise COBOL and PL/I” on page 64<br>• Chapter 4, “Host-based code analysis,” on page 57 |
| CICS TS update | none | “xUnit support for CICS applications (ZUnit)” on page 65 |

The configuration of Developer for z/OS requires more than the typical system programming permissions and expertise; therefore, assistance from others might be needed. Table 3 on page 6 and Table 4 on page 6 list the administrators who are needed for the required and optional customization tasks.

| Table 3. Administrators needed for required tasks | | |
|---|---|---|
| **Administrator** | **Task** | **Information** |
| System | Typical system programmer actions are required for all customization tasks | N/A |

| Table 4. Administrators needed for optional tasks | | |
|---|---|---|
| **Administrator** | **Task** | **Information** |
| Security | Define data set profiles | "Security considerations" in *Host Configuration Reference* |
| TCP/IP | Define new TCP/IP ports | • “CRASRV.properties, the RSE interface to CARMA” on page 36<br>• “Multiple Code coverage invocations” on page 58<br>• “xUnit support for CICS applications (ZUnit)” on page 65 |
| CICS TS | • Update CICS region JCL<br>• Update CICS region CSD<br>• Update CICS region SIP<br>• Define CICS resources | • “CICS bidirectional language support” on page 69<br>• “xUnit support for CICS applications (ZUnit)” on page 65 |
| WLM | Assign goals to Developer for z/OS tasks | "WLM considerations" in the *Host Configuration Reference* |

# Preconfiguration considerations

For information about Developer for z/OS itself, how it interacts with your system, and with the prerequisite and co-requisite products, see the *Developer for z/OS Host Configuration Reference*. The *IBM Explorer for z/OS Host Configuration Reference* gives similar information for z/OS Explorer, which is a requisite for Developer for z/OS. This information can assist you in creating a setup that supports your current needs and future growth.

Before configuring Developer for z/OS, ensure you configured z/OS Explorer.

## Product selection

The host components of Developer for z/OS are used by different products. When in doubt, contact the person responsible for the purchase of FMID HHOPxxx ( z/OS Explorer Extensions), or possibly your local IBM representative, to learn which product was purchased. With this information, you can select the correct product registration method, as pricing and available features differ for each of the following products, and the host components will register only as one of the possible products.

**IBM Wazi for Red Hat CodeReady Workspaces (program number 5900-A8N)**
The Wazi client enhances the functionality provided by z/OS Explorer. Activation of client features is managed by the client.

**IBM Developer for z/OS (program number 5724-T07)**
The Developer for z/OS client enhances the functionality of the Eclipse client provided by provided by IBM Wazi for Red Hat CodeReady Workspaces. Activation of client features is managed by the client.

**IBM Developer for z/OS Enterprise Edition (program number 5755-AC5)**
The Developer for z/OS Enterprise Edition provides the same Eclipse client functionality as Developer for z/OS. However, activation of client features is managed by the host, and this product provides additional debug related capabilities, such as a 3270 interface, and additional clients.

**IBM Application Delivery Foundation for z Systems® (program number 5655-AC6)**
Application Delivery Foundation for z/OS provides Developer for z/OS Enterprise Edition, which is described earlier, combined with other products useful for z/OS application development activities.

## Workload management

Unlike traditional z/OS applications, Developer for z/OS is not a monolithic application that can be identified easily to Workload Manager (WLM). Developer for z/OS consists of several components that interact to give the client access to the host system services and data. To plan your WLM configuration, see "WLM considerations" in the *Host Configuration Reference*.

**Note:** Developer for z/OS consists of multiple tasks that communicate with each other and the client. These tasks use various timers to detect communication loss with their partners. Timeout issues can arise (due to lack of CPU time during the timeout window) on systems with a heavy CPU load or incorrect Workload Management (WLM) settings for Developer for z/OS.

## Resource usage and system limits

Developer for z/OS uses a variable number of system resources such as address spaces, and z/OS UNIX processes and threads. The availability of these resources is limited by various system definitions. To estimate the usage of key resources so that you can plan your system configuration, see "Tuning considerations" in the *IBM Explorer for z/OS Host Configuration Reference*. Developer for z/OS can run in either 31-bit or 64-bit mode, changing the storage resource limitations drastically.

## Required configuration of requisite products

Consult your MVS system programmer, security administrator, and TCP/IP administrator to verify if the requisite products and software are installed, tested, and working. Some requisite customization tasks that can be overlooked are listed here:

- All Developer for z/OS users must have READ and EXECUTE access to the Java directories.
- Ensure that at least the basic configuration of z/OS Explorer has completed, since Developer for z/OS customization extends this setup.
- Remote (host-based) actions for z/OS UNIX subprojects require that z/OS UNIX version of REXEC or SSH is active on the host system.

## User ID considerations

The user ID of a Developer for z/OS user must have at least the following attributes:

- TSO access (with a normal region size).

  **Note:** A large region size is required for the user ID that runs the Installation Verification Programs (IVPs) because functions requiring a lot of memory (such as Java) are executed. You should set the region size to 131072 kilobytes (128 megabytes) or higher.
- An OMVS segment defined to the security system (for example, RACF), both for the user ID and its default group.

- The HOME field must refer to a home directory allocated for the user (with READ, WRITE and EXECUTE access).
- The PROGRAM field in the OMVS segment should be `/bin/sh` or other valid z/OS UNIX shell, such as `/bin/tcsh`.
- The ASSIZEMAX field should not be set, so that system defaults are used.
- The user ID does not require UID 0.

  Example (command **`LISTUSER userid NORACF OMVS`**):

```
USER=userid

OMVS INFORMATION
----------------
UID= 0000003200
HOME= /u/userid
PROGRAM= /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMAX= NONE
PROCUSERMAX= NONE
THREADSMAX= NONE
MMAPAREAMAX= NONE
```

- The user ID's default group requires a `GID`.

  Example (command **`LISTGRP group NORACF OMVS`**):

```
GROUP group

OMVS INFORMATION
----------------
GID= 0000003243
```

- READ and EXECUTE access to the z/OS Explorer and Developer for z/OS installation and configuration directories and files, default `/usr/lpp/IBM/zexpl/*`, `/etc/zexpl/*`, `/var/zexpl/*`, `/usr/lpp/IBM/idz/*`, and `/var/idz/*`.
- READ, WRITE, and EXECUTE access to the z/OS Explorer WORKAREA directory, default `/var/zexpl/WORKAREA`, and user log directory, default `/var/zexpl/logs`.
- READ access to the z/OS Explorer and Developer for z/OS installation data sets, default `FEK.SFEK*` and `FEL.SFEL*`.
- READ, WRITE, and EXECUTE access to the `/tmp` directory or a directory referenced in the `TMPDIR` environment variable.

## Server considerations

z/OS Explorer and Developer for z/OS consists of multiple permanently active servers, which can be started tasks or user jobs. These servers provide the requested services themselves or start other servers (as z/OS UNIX threads or user jobs) to provide the service. There is no specific startup order.

The only requirement is that the servers are up and running before the first user tries to connect. The security mechanisms used by z/OS Explorer and Developer for z/OS servers and services rely on the data sets and file systems they reside in being secure. This implies that only trusted system administrators should be able to update the program libraries and configuration files.

# Predeployment considerations

Developer for z/OS supports the cloning of an installation to a different system, thus avoiding the need for a SMP/E installation on each system.

The following data sets, directories, and files are mandatory for deployment to other systems. If you copied a file to a different location, this file must replace its counterpart in the following lists.

**Note:** The following list does not cover the deployment needs of the prerequisite and co-requisite software (including z/OS Explorer).

z/OS Explorer Extensions

- FEL.SFELLMOD(*)
- FEL.SFELLOAD(*)
- FEL.SFELLPA(*)
- FEL.SFELPROC(*)
- FEL.#CUST.CNTL(*)
- FEL.#CUST.PARMLIB(*)
- FEL.#CUST.PROCLIB(*)
- /usr/lpp/IBM/zee/*
- /etc/zexpl/*
- definitions, data sets, files, and directories resulting from customization jobs in FEL.#CUST.JCL

z/OS Source Code Analysis

- AKG.SAKGPROC(*)
- /usr/lpp/IBM/akg/*

**Notes:**

- FEL and `/usr/lpp/IBM/zee` are the high-level qualifier and path used during the installation of the z/OS Explorer Extensions. FEL.#CUST and `/etc/zexpl` are the default locations used during the customization of the product where `/etc/zexpl` is the z/OS Explorer configuration directory.
- AKG and `/usr/lpp/IBM/akg` are the high-level qualifier and path used during the installation of z/OS Source Code Analysis.
- You should install z/OS Explorer Extensions in a private file system, possibly combined with z/OS Explorer, to ease the deploying of the z/OS UNIX parts of the product. If you cannot use a private file system, use an archiving tool such as the z/OS UNIX `tar` command to transport the z/OS UNIX directories from one system to another. This method is required for preserving the attributes (such as program control) of the Developer for z/OS files and directories.

  For more information about the following sample commands to archive and restore the Developer for z/OS installation directory, see *UNIX System Services Command Reference* (SA22-7802).

  - Archive: `cd /SYS1/usr/lpp/IBM/zee; tar -cSf /u/userid/zee.tar`
  - Restore: `cd /SYS2/usr/lpp/IBM/zee; tar -xSpf /u/userid/zee.tar`

## Client checklist

Users of the Developer for z/OS client must know the result of certain host system customizations, such as TCP/IP port numbers, for the client to work properly. Use these checklists to gather the information needed.

The checklist in Table 5 on page 9 lists the required results of mandatory customization steps. Table 6 on page 10 lists the required results of optional customization steps.

*Table 5. Client checklist: Mandatory parts*

| **Customization** | **Value** |
|---|---|
| (prerequisite) RSE daemon TCP/IP port number. The default is 4035.<br>This port is set during z/OS Explorer configuration. | |

| *Table 6. Client checklist: Optional parts* | |
|---|---|
| **Customization** | **Value** |
| Location of the ELAXF* procedures if they are not in a system procedure library. The default is FEL.#CUST.PROCLIB.<br>See the note on JCLLIB in "ELAXF* remote build procedures" on page 19. | |
| Procedure or step names of the ELAXF* procedures if they were changed.<br>See the note on changing them in "ELAXF* remote build procedures" on page 19. | |
| Location of the AKGCR procedure if it is not in a system procedure library. The default is AKG.#CUST.PROCLIB.<br>See the note on JCLLIB in "Code review" on page 57. | |
| Location of the AKGCC procedure if it is not in a system procedure library. The default is AKG.#CUST.PROCLIB.<br>See the note on JCLLIB in "Code coverage" on page 58. | |
| Location of the FEKRNPLI Include Preprocessor exec statement. The default is FEL.#CUST.CNTL.<br>See "Include preprocessor support" on page 63. | |
| Location of the FELPACK pack/unpack program, if it is not in a system procedure library. The default is FEL.SFELPROC.<br>See ISPF PACK support. | |
| Location of the unit test load modules if not in LINKLIST or STEPLIB of zee.env. The default is FEL.SFELLOAD.<br>See "xUnit support for Enterprise COBOL and PL/I" on page 64. | |
| Location of the AZUZUNIT procedure if it is not in a system procedure library. The default is FEL.#CUST.PROCLIB.<br>See the note on JCLLIB in "xUnit support for Enterprise COBOL and PL/I" on page 64. | |
| Location of the sample *.xsd and *.xsl XML files used for unit test output formatting. The defaults are /usr/lpp/IBM/zee/samples/zunit/xsd and /usr/lpp/IBM/zee/samples/zunit/xsl.<br>See "xUnit support for Enterprise COBOL and PL/I" on page 64. | |
| TCP/IP port number for unit test CICS recorder. There is no default.<br>See "xUnit support for CICS applications (ZUnit)" on page 65. | |
| (co-requisite) TN3270 port number for Host Connect Emulator. The default is 23.<br>See "TCP/IP ports" in *Host Configuration Reference*. | |
| (co-requisite) REXEC or SSH port number, which, by default are 512 or 22.<br>See "z/OS UNIX subprojects" on page 62. | |
| Location of the SFELSAMP sample library for CARMA RAM samples. The default is FEL.SFELSAMP.<br>See the *CARMA Developer's Guide* (SC23-7660). | |

| *Table 6. Client checklist: Optional parts (continued)* | |
|---|---|
| **Customization** | **Value** |
| Location of the CRA#ASLM JCL for CARMA SCLM RAM data set allocations. The default is FEL.#CUST.JCL.<br>See the note on CRA#ASLM in "SCLM RAM" on page 45. | |

# Chapter 2. Basic customization

The following customization steps are common for the various Developer for z/OS services. See the chapters about the specific components for their customization requirements.

## Requirements and checklist

You need the assistance of a security administrator and a TCP/IP administrator to complete this customization task, which requires the following resources and special customization tasks:

- LPA data set
- Various PARMLIB updates
- Various PROCLIB updates
- Various security software updates
- Various TCP/IP ports for internal and client-host communication

To verify the installation and to start using Developer for z/OS at your site, do the following tasks. Unless otherwise indicated, all tasks are mandatory.

1. Create customizable copies of samples and create the work environment for Developer for z/OS. For details, see "Customization setup" on page 13.
2. Update SYS1.PARMLIB definitions. For details, see "PARMLIB changes" on page 14.
3. Create SYS1.PROCLIB procedures. For details, see "PROCLIB changes" on page 19.
4. Update security definitions. For details, see "Security definitions" on page 21.
5. Customize z/OS Explorer configuration files. For details, see "rse.env, the z/OS Explorer configuration file" on page 22.
6. Customize Developer for z/OS configuration files. For details, see "zee.env, the environment configuration file" on page 22.

## Customization setup

Developer for z/OS contains several sample configuration files and sample JCL. To avoid overwriting your customizations when applying maintenance, copy all of these members and z/OS UNIX files to a different location, and customize the copy.

The Developer for z/OS FMIDs are shared by multiple IBM products, so it is possible you see configurable files, libraries, or directories that are not applicable to your setup. When in doubt, refer back to this document, as it contains all information on the parts that do apply to your setup.

To create customizable copies of configuration files and configuration JCL, and to create required z/OS UNIX directories, customize and submit the sample FELSETUP member in the FEL.SFELSAMP data set. The required customization steps are described within the member.

This job performs the following tasks:

- Create FEL.#CUST.PARMLIB and populate it with sample configuration files.
- Create FEL.#CUST.PROCLIB and populate it with sample SYS1.PROCLIB members.
- Create FEL.#CUST.JCL and populate it with sample configuration JCL.
- Create FEL.#CUST.CNTL and populate it with sample server startup scripts.
- Create FEL.#CUST.ASM and populate it with sample assembler source code.
- Create FEL.#CUST.SQL and populate it with sample SQL command files.
- Populate the z/OS Explorer configuration directory, /etc/zexpl/* with sample configuration files.

© Copyright IBM Corp. 2000, 2024

**Notes:**

- The customization steps described here, including the FELSETUP tool, assume that the configuration of z/OS Explorer has already been completed.
- The configuration steps in this publication use the member and file locations created by the FELSETUP job, unless noted otherwise. The original samples, which should not be updated, are in FEL.SFELSAMP and /usr/lpp/IBM/zee/samples/.
- For more details on which sample members are copied to which data set, and for more details on which directories are created, their permission bitmask, and where the various sample files are copied to, see the comments in FEL.SFELSAMP(FELSETUP).
- To aid in migrating an existing setup, the comments in FEL.SFELSAMP(FELSETUP) also document the changes between different versions of Developer for z/OS.

# PARMLIB changes

The following PARMLIB changes are documented in this section:



For more information about the PARMLIB definitions listed in the next sections, see *MVS Initialization and Tuning Reference* (SA22-7592). For more information about the sample console commands, see *MVS System Commands* (SA22-7627).

## Set the z/OS UNIX definitions in BPXPRMxx

Define OMVS=xx in the IEASYSxx parmlib member to specify which BPXPRMxx parmlib member should be used during IPL.

MAXPROCUSER specifies the maximum number of processes that a single z/OS UNIX user ID can have concurrently active. Set MAXPROCUSER in SYS1.PARMLIB(BPXPRMxx) to 50 or higher. This setting is intended to be a system-wide limit, because it should be active for each client that uses Developer for z/OS.

These values can be checked and set dynamically (until the next IPL) with the following console commands:

- DISPLAY OMVS,O
- SETOMVS MAXPROCUSER=50

**Note:**

- The MAXPROCUSER value suggested here is based upon users having a unique z/OS UNIX user ID (UID). Increase this value if your users share the same UID.
- During the SMP/E install of z/OS Explorer Extensions, you were advised to place the code in a separate file system (zFS of HFS) and update BPXPRMxx to mount this file system during system IPL. Included is a repeat of the sample mount command in case this update still must be done:

```
MOUNT FILESYSTEM('#dsn')
   MOUNTPOINT('-PathPrefix-usr/lpp/IBM/zee')
   SETUID                     /* preserve APF auth */
   MODE(RDWR)                 /* can be MODE(READ) */
   TYPE(ZFS) PARM('AGGRGROW') /* zFS, with extents */
/* TYPE(HFS) */               /* HFS, auto. extent */
```

- During the SMP/E install of z/OS Source Code Analysis, you were advised to place the code in a separate file system (zFS of HFS) and update BPXPRMxx to mount this file system during system IPL. Included is a repeat of the sample mount command in case this update still must be done:

```
MOUNT FILESYSTEM('#dsn')
   MOUNTPOINT('-PathPrefix-usr/lpp/IBM/akg')
   MODE(RDWR)                /* can be MODE(READ) */
   TYPE(ZFS) PARM('AGGRGROW') /* zFS, with extents */
/* TYPE(HFS) */              /* HFS, auto. extent */
```

## Product enablement in IFAPRDxx

The host components of Developer for z/OS are used by different products. When in doubt, contact the person responsible for the purchase of FMID HHOPxxx ( z/OS Explorer Extensions), or possibly your local IBM representative, to learn which product was purchased. With this information, you can select the correct product registration method, as pricing and available features differ for each of the following products, and the host components will register only as one of the possible products.

**IBM Wazi for Red Hat CodeReady Workspaces (program number 5900-A8N)**
The Wazi client enhances the functionality provided by z/OS Explorer. Activation of client features is managed by the client.

**IBM Developer for z/OS (program number 5724-T07)**
The Developer for z/OS client enhances the functionality of the Eclipse client provided by provided by IBM Wazi for Red Hat CodeReady Workspaces. Activation of client features is managed by the client.

**IBM Developer for z/OS Enterprise Edition (program number 5755-AC5)**
The Developer for z/OS Enterprise Edition provides the same Eclipse client functionality as Developer for z/OS. However, activation of client features is managed by the host, and this product provides additional debug related capabilities, such as a 3270 interface, and additional clients.

**IBM Application Delivery Foundation for z/OS (program number 5655-AC6)**
Application Delivery Foundation for z/OS provides Developer for z/OS Enterprise Edition, which is described earlier, combined with other products useful for z/OS application development activities.

Products to be enabled on z/OS are defined in SYS1.PARMLIB(IFAPRDxx). Define PROD=xx in the IEASYSxx parmlib member to specify which IFAPRDxx parmlib member should be used during IPL.

The Developer for z/OS host components will attempt to register using the following order of product definitions. The process stops on the first successful registration.

Specify the following in IFAPRDxx to define IBM Application Delivery Foundation for z/OS (product code 5655-AC6):

```
PRODUCT OWNER('IBM CORP')
        NAME('IBM APP DLIV FND')
        ID(5655-AC6)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(*)
        STATE(ENABLED)
```

Specify the following in IFAPRDxx to define IBM Developer for z/OS Enterprise Edition (product code 5755-AC5):

```
PRODUCT OWNER('IBM CORP')
        NAME('IBM IDz EE')
        ID(5755-AC5)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(*)
        STATE(ENABLED)
```

Specify the following in IFAPRDxx to define IBM Developer for z/OS (product code 5724-T07):

```
PRODUCT OWNER('IBM CORP')
        NAME('IBM IDz')
        ID(5724-T07)
        VERSION(*) RELEASE(*) MOD(*)
```

```
        FEATURENAME(*)
        STATE(ENABLED)
```

Specify the following in IFAPRDxx to define IBM Wazi for Red Hat CodeReady Workspaces (product code 5900-A8N):

```
PRODUCT OWNER('IBM CORP')
       NAME('IBM Wazi Code')
       ID(5900-A8N)
       VERSION(*) RELEASE(*) MOD(*)
       FEATURENAME(*)
       STATE(ENABLED)
```

After the IFAPRDxx parmlib member is updated, it can be activated dynamically (until the next IPL) with the following console command:

```
SET PROD=xx
```

**Note:** All product flavors of the host components register the following features:

- FEL-RSED (for RSE daemon in z/OS Explorer Extensions, FMID HHOPxxx)
- AKG-CC (for Code Coverage in z/OS Source Code Analysis, FMID HAKGxxx)
- AKG-CR (for Code Review in z/OS Source Code Analysis, FMID HAKGxxx)

A product that is not defined in IFAPRDxx, or defined with STATE(DISABLED) or STATE(NOTDEFINED) will not be selected for registration. If none of the documented products is defined in IFAPRDxx, z/OS Explorer Extensions will only provide basic services for clients with a valid activation token.

If you change how a product is purchased, for example, you upgrade from using a stand-alone version to the IBM Application Delivery Foundation for z Systems product bundle, you must explicitly remove the existing product definition from the in-storage tables kept by z/OS when activating the new definition. Follow this scenario to do this dynamically (without IPL):

1. In IFAPRDxx, define the new product as described before, and update the old product with STATE(DISABLED).
2. Activate the update with operator command SET PROD=xx.
3. Restart the RSED server to pick up the change.
4. You can now safely remove the old product definition from IFAPRDxx.

IBM advises against defining IFAPRDxx entries that have NAME(*) or ID(*) fields, because this will result in ALL z/OS applications that utilize product registration to find a match on the first test, and adhere to the related STATE() definition. For z/OS Explorer Extensions with STATE(ENABLED), this means that the application will register as *IBM Application Delivery Foundation for z Systems* (product code 5655-AC6).

## LPA definitions in LPALSTxx

The optional Common Access Repository Manager (CARMA) service supports different server startup methods for the CARMA server. The CRASTART startup method requires that the modules in the FEL.SFELLPA load library are in the Link Pack Area (LPA).

LPA data sets are defined in SYS1.PARMLIB(LPALSTxx). Define LPA=xx in the IEASYSxx parmlib member to specify which LPALSTxx parmlib member should be used during IPL.

LPA definitions can be set dynamically (until the next IPL) with the following console command:

- SETPROG LPA,ADD,DSN=FEL.SFELLPA,MASK=*

**Note:**

- Data sets listed in LPALSTxx must be cataloged in the master catalog or a user catalog identified in the LPALSTxx member.
- Adding a new data set to LPALSTxx requires an IPL with CLPA (create LPA) to be activated.

- All libraries that are loaded into LPA are automatically considered to be APF-authorized and program controlled. Ensure you have proper security controls in place for these libraries.
- If you choose to not place a library designed for LPA placement in LPA and you use LINKLIST or STEPLIB instead, ensure that you define the APF authorization and program control status.

## LINKLIST definitions in PROGxx

LINKLIST definitions for Developer for z/OS can be grouped in three categories:

- Developer for z/OS load libraries that are needed for Developer for z/OS functions. These definitions are described in this section.
- Requisite load libraries that are needed for Developer for z/OS functions. These definitions are described in “Requisite LINKLIST and LPA definitions ” on page 18.
- Developer for z/OS load libraries that are needed by other products. These definitions are described in “LINKLIST definitions for other products” on page 19.

*Table 7. Match load modules to functions.* This table describes the functions of load modules and the load libraries where they are located.

| Load library | Load modules | Usage | STEPLIB |
|---|---|---|---|
| FEL.SFELLMOD | IRZ* and IIRZ* | “Diagnostic IRZ messages for Enterprise Service Tools” on page 70 | CICS, IMS, or MVS batch |
| FEL.SFELLOAD | AZU* and IAZU* | “xUnit support for Enterprise COBOL and PL/I” on page 64 | zee.env or MVS batch |
| | CRA* | Chapter 3, “Common Access Repository Manager (CARMA),” on page 27 | CRASUB* or crastart*.conf |
| | ELAX* | “ELAXF* remote build procedures” on page 19<br>(error feedback and include preprocessor) | ELAXF* procedures |
| | FEJB* | “CICS bidirectional language support” on page 69 | CICS |
| FEL.SFELLPA | CRA* | Chapter 3, “Common Access Repository Manager (CARMA),” on page 27 | CRASRV.properties |

In order for the listed Developer for z/OS services to work, all modules documented in Table 7 on page 17 that are related to the service must be made available either through STEPLIB or LINKLST (or LPA). Note that the SFELLMOD library is not used by Developer for z/OS itself, but by code generated by Developer for z/OS. See the STEPLIB column in Table 7 on page 17 if you choose to use STEPLIB to learn where the STEPLIB (or DFHRPL for CICS) definition must be made. However, you should be aware of the following things:

- Using STEPLIB in z/OS UNIX has a negative performance impact.
- If one STEPLIB library is APF-authorized, then all must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.
- Libraries added to the STEPLIB DD in a JCL are not propagated to the z/OS UNIX processes started by the JCL.

LINKLIST data sets are defined in SYS1.PARMLIB(PROGxx), if your site followed IBM recommendations. Define PROG=xx in the IEASYSxx parmlib member to specify which PROGxx parmlib member should be used during IPL.

The required definitions will look like the following, where `listname` is the name of the LINKLIST set that will be activated, and `volser` is the volume on which the data set resides if it is not cataloged in the master catalog:

- `LNKLST ADD NAME(listname) DSNAME(FEL.SFELLMOD) VOLUME(volser)`
- `LNKLST ADD NAME(listname) DSNAME(FEL.SFELLOAD)`

LINKLIST definitions can be created dynamically (until the next IPL) with the following actions:

1. Create a new PARMLIB member named `PROGxx`, for example `PROGLL`.
2. Place the following commands in your new `PROGxx` member, where `volser` is the volume on which the data set resides if it is not cataloged in the master catalog:
   - `LNKLST DEFINE,NAME=LLTMP,COPYFROM=CURRENT`
   - `LNKLST ADD NAME=LLTMP,DSN=FEL.SFELLMOD,VOL=volser`
   - `LNKLST ADD NAME=LLTMP,DSN=FEL.SFELLOAD`
   - `LNKLST ACTIVATE,NAME=LLTMP`
3. Activate the new definitions by issuing the **SET PROG=xx** console command, where `xx` matches the last 2 characters of your new `PROGxx` member name.

## Requisite LINKLIST and LPA definitions

The following additional libraries must be made available, either through STEPLIB or LINKLIST/LPA, to support the use of optional services. This list does not include data sets that are specific to a product that Developer for z/OS interacts with, like z/OS Explorer:

- System load library
  - `SYS1.LINKLIB`
- Language Environment® runtime
  - `CEE.SCEERUN`
  - `CEE.SCEERUN2`
- C++'s DLL class library
  - `CBC.SCLBDLL`
- System load library (for Enterprise COBOL and PL/I unit test)
  - `SYS1.CSSLIB`
  - `SYS1.SIXMLOD1`

**Note:**

- All libraries that are loaded into LPA are automatically considered to be APF-authorized and program controlled. Ensure you have proper security controls in place for these libraries.
- Libraries that are designed for LPA placement, such as `REXX.*.SEAGLPA`, might require additional program control or APF authorizations if they are accessed through LINKLIST or STEPLIB.
- Some of the prerequisite and co-requisite products, such as z/OS Explorer, also require STEPLIB or LINKLIST/LPA definitions. See the related product customization guides for more information.

LINKLIST data sets are defined in `SYS1.PARMLIB(PROGxx)` by default. LPA data sets are defined in `SYS1.PARMLIB(LPALSTxx)`.

If you opt to use STEPLIB, you must define the libraries not available through LINKLIST/LPA in the STEPLIB directive of `rse.env`, the RSE configuration file. Be aware, however, of these things:

- Using STEPLIB in z/OS UNIX has a negative performance impact.
- If one STEPLIB library is APF-authorized, then all the other STEPLIB libraries must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.

- Libraries added to the STEPLIB DD in a JCL are not propagated to the z/OS UNIX processes started by the JCL.

## LINKLIST definitions for other products

The Developer for z/OS client has a code generation component called Enterprise Service Tools. In order for the generated code to issue diagnostic error messages, all IRZM* and IIRZ* modules in the FEL.SFELLMOD load library must be made available either through STEPLIB or LINKLIST.

LINKLIST data sets are defined in SYS1.PARMLIB(PROGxx) by default.

If you opt to use STEPLIB, you must define the libraries that are not available through LINKLIST in the STEPLIB directive of the task that executes the code (IMS or batch job). However, if one STEPLIB library is APF-authorized, then all other STEPLIB libraries must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.

## SMF record collection in SMFPRMxx

System Management Facilities (SMF) uses SYS1.PARMLIB(SMFPRMxx) to determine which record types and subtypes should be collected. Define SMF=xx in the IEASYSxx parmlib member to specify which SMFPRMxx parmlib member should be used during IPL.

Developer for z/OS creates SMF records type 122, subtype 1. Collection of these records is required if you want to use the Developer for z/OS Enterprise Edition host to activate all Developer for z/OS client functions.

Update the SYS(TYPE()) definition in SMFPRMxx to collect type 122 records.

Updated SMFPRMxx definitions can be activated dynamically (until the next IPL) with the following console command:

```
SET SMF=xx
```

# PROCLIB changes

The following PROCLIB changes are documented in this section:


- "ELAXF* remote build procedures" on page 19


The remote build procedures listed in the following sections must reside in a system procedure library defined to your JES subsystem. In the instructions in the following sections, the IBM default procedure library, SYS1.PROCLIB, is used.

## ELAXF* remote build procedures

Developer for z/OS provides sample JCL procedures that can be used for the JCL generation, remote project builds, and remote syntax check features of CICS BMS maps, IMS MFS screens, Assembler, C/C++, COBOL and PL/I programs. These procedures allow installations to apply their own standards, and ensure that developers use the same procedures with the same compiler options and compiler levels.

The sample procedures and their function are listed in Table 8 on page 19.

*Table 8. Sample ELAXF* procedures*

| Member | Purpose |
|---|---|
| ELAXF | Include member that specifies variables used by the remote build procedures. |
| ELAXFADT | Sample procedure for assembling and debugging High Level assembler programs. |
| ELAXFASM | Sample procedure for assembling High Level assembler programs. |
| ELAXFBMS | Sample procedure for creating CICS BMS object and corresponding copy, dsect, or include member. (3) |

| *Table 8. Sample ELAXF* procedures (continued)* | |
|---|---|
| **Member** | **Purpose** |
| ELAXFCOC | Sample procedure for COBOL compiling and doing Integrated CICS translate and integrated Db2® translate. (1) |
| ELAXFCOP | Sample procedure for Db2 preprocessing of EXEC SQL statements embedded in COBOL programs. |
| ELAXFCOT | Sample procedure for CICS translation for EXEC CICS statements embedded in COBOL programs. |
| ELAXFCPC | Sample procedure for C compiling. |
| ELAXFCPP | Sample procedure for C++ compiling. |
| ELAXFCP1 | Sample procedure for COBOL compiling with SCM preprocessor statements (-INC and ++INCLUDE). (1) |
| ELAXFDCL | Sample procedure for running a program in TSO mode. |
| ELAXFGO | Sample procedure for the GO step. |
| ELAXFLNK | Sample procedure for linking C/C++, COBOL, PL/I, and High Level Assembler programs. (1) (3) |
| ELAXFMFS | Sample procedure for creating IMS MFS screens. |
| ELAXFPLP | Sample procedure for Db2 preprocessing of EXEC SQL statements embedded in PLI programs. |
| ELAXFPLT | Sample procedure for doing CICS translation of EXEC CICS statements embedded in PLI programs. |
| ELAXFPL1 | Sample procedure for PL/I compiling, and integrated CICS translation and integrated Db2 translation. (1) |
| ELAXFPP1 | Sample procedure for PL/I compiling with SCM preprocessor statements (-INC and ++INCLUDE). (1) |
| ELAXFSP | Sample procedure to register a stored procedure to Db2. (2) |
| ELAXFSQL | Sample procedure to invoke SQL. (2) |
| ELAXFTSO | Sample procedure for running and debugging the generated Db2 code in TSO mode. |
| ELAXFUOP | Sample procedure for generating the UOPT step when building programs that run in CICS or IMS subsystems. |

All remote build procedures rely on the ELAXF include member for the definition of common high-level qualifiers. Customize the sample include member, `FEL.#CUST.PROCLIB(ELAXF)` as described within the member. You can use Table 9 on page 21 and Table 10 on page 21 to assist with this customization.

Review the sample build procedure members, `FEL#CUST.PROCLIB(ELAXF*)`, and customize them if required. Some sample `ELAXF*` procedures in Table 8 on page 19 are marked with a footnote as they are more likely to require customization:

1. These remote build procedures have commented out references to CICS and Db2 load libraries.
2. These remote build procedures have a reference to a customized input file with SQL commands.
3. These remote build procedures have miscellaneous customizable options.

The names of the procedures and the names of the steps in the procedures match the default properties that are included with the Developer for z/OS client. If the name of a procedure or the name of a step in a procedure is changed, the corresponding properties file on all of the clients must be updated. You should not change the procedure and step names.

| *Table 9. ELAXF high-level qualifier checklist* | | |
|---|---|---|
| **Product** | **Default HLQ** | **Value** |
| Developer for z/OS | FEL | |
| Debugger | EQAW | |
| ADFz Common Components | IPV | |
| COBOL | IGY.V6R2M0 | |
| PL/I | PLI.V5R2M0 | |
| C/C++ | CBC | |
| LE | CEE | |
| XML Toolkit | SYS1 | |
| CICS | CICSTS54.CICS | |
| Db2 | DSNA12 | |
| IMS (site specific libraries) | IMS | |
| system LINKLIB | SYS1 | |
| system MACLIB | SYS1 | |
| system CSSLIB | SYS1 | |

Some ELAXF* procedures reference data set names that do not have fixed low-level qualifiers. An example is the Db2 runtime library, which holds Db2 utilities that are compiled by your Db2 administrator. Use Table 10 on page 21 to map the default data set names to the names used at your site.

| *Table 10. ELAXF*. fully qualified data set checklist | | |
|---|---|---|
| **Product** | **Default DSN** | **Value** |
| Developer for z/OS - SQL samples | FEL.#CUST.SQL | |
| Db2 runtime libraries | DSNA12.RUNLIB.LOAD | |

Once the ELAXF* members are customized, copy them to SYS1.PROCLIB. Ensure that the ELAXF include member remains with the ELAXF* build procedures. If the members cannot be copied into a system procedure library, ask the Developer for z/OS users to add a JCLLIB card (right after the JOB card) to the job properties on the client.

```
//MYJOB    JOB <job parameters>
//PROCS    JCLLIB ORDER=(FEL.#CUST.PROCLIB)
```

## Security definitions

To create the security definitions for Developer for z/OS, customize and submit the sample FELRACF member. The user submitting this job must have security administrator privileges, such as being RACF SPECIAL.

FELRACF is located in FEL.#CUST.JCL, unless you specified a different location when you customized and submitted the FEL.SFELSAMP(FELSETUP) job. For more details, see "Customization setup" on page 13.

The following list of security-related definitions for Developer for z/OS are discussed in detail in Chapter 7, "Security definitions," on page 73.

- Define data set profiles
- Verify the security settings

## rse.env, the z/OS Explorer configuration file

The z/OS Explorer RSE server processes (RSE daemon, RSE thread pool, and RSE server) must know that they are extended by z/OS Explorer Extensions. Depending on where z/OS Explorer Extensions was installed, this detection is automatic, or done via an environment variable in `rse.env`.

If z/OS Explorer Extensions was installed in the `plugin/` directory of z/OS Explorer, default `/usr/lpp/IBM/zexpl/plugin/`, then z/OS Explorer automatically detects the presence of z/OS Explorer Extensions on the next restart of the RSED started task, and no further action is required.

If z/OS Explorer Extensions was installed in another directory (the default is `/usr/lpp/IBM/zee/`), then variable `RSE_PLUGIN_PATH` must be updated in `rse.env` to specify the install location of z/OS Explorer Extensions.

`rse.env` is located in `/etc/zexpl/`, unless you specified a different location when you customized and submitted the z/OS Explorer `FEK.SFEKSAMP(FEKSETUP)` job. You can edit the file with the TSO **OEDIT** command. For more information, see "Customization setup" on page 13.

**Note:**

- For your changes to take effect, the z/OS Explorer RSED started task must be restarted.
- Usage of `RSE_PLUGIN_PATH` requires that z/OS Explorer 3.2 PTF `UI71761` is applied.

```
#RSE_PLUGIN_PATH=
```

*Figure 1. rse.env: RSE_PLUGIN_PATH variable*

**RSE_PLUGIN_PATH**

Reference to plugin products that are not installed in the `plugin/` directory. The default is an empty string. Uncomment and change to match the installation paths of products that extend z/OS Explorer. Multiple paths are separated by a semicolon (:).

Products that are installed in the `plugin/` sub-directory are detected automatically and do not need to be added to `RSE_PLUGIN_PATH`.

For example: `RSE_PLUGIN_PATH=/usr/lpp/IBM/zee`

## zee.env, the environment configuration file

The z/OS Explorer RSE server processes (RSE daemon, RSE thread pool, and RSE server) use the definitions in `zee.env` to learn about z/OS Explorer Extensions environment variables.

`zee.env` is located in `/etc/zexpl/`, unless you specified a different location when you customized and submitted the `FEL.SFELSAMP(FELSETUP)` job. For more details, see "Customization setup" on page 13. You can edit the file with the TSO **OEDIT** command.

See the following sample `zee.env` file, which can be customized to match your system environment. Default values are provided for all variables that are not explicitly specified. The syntax of the file follows standard z/OS UNIX shell syntax rules. For example, comments start with a number sign (#) when using a US code page, and spaces around the equal sign (=) are not supported.

**Note:** For your changes to take effect, the z/OS Explorer RSED started task must be restarted.

```
ELAXF=
BZUINCL=
# convert SET statements in INCLUDE to ${incl}_${symbol} envvars
. $FEL_HOME/bin/felSetToEnvvar ELAXF $ELAXF
. $FEL_HOME/bin/felSetToEnvvar BZUINCL $BZUINCL
```

```
FEL_HLQ=${ELAXF_FEL:-FEL}
```

```
## debug
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDISABLE_DBM_INTEGRATION=false"
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Ddebug.miner.autoreconnect=0"
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Ddebug.miner.localhost=localhost"
```

```
##  C/C++
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dinclude.c=/etc/zexpl/include.conf"
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dinclude.cpp=/etc/zexpl/include.conf"
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DCPP_CLEANUP_INTERVAL=60000"
```

```
## remote index search
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DRIS_BUFFER=8"
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDISABLE_REMOTE_INDEX_SEARCH=true"
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDEFAULT_REMOTE_ZOS_FILE_SEARCH=true"

## system
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDISABLE_DELETE_IN_SUBPROJECT=true"
#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dsclm.check.enable=true"
```

```
## ISPF EDIT macro
#FEL_EDIT_MACRO_DSN=$FEL_HLQ.SFELSAMP
#FEL_EDIT_MACRO=FELEDTMC
```

```
## ZUnit unit test
#STEPLIB=$STEPLIB:$FEL_HLQ.SFELLOAD
#STEPLIB=$STEPLIB:SYS1.CSSLIB:SYS1.SIXMLOD1
#AZU_JES_PROCLIB=SYS1.PROCLIB
```

```
## EWM (RTC) user build
#FEL_UBLD_DD=$CGI_ISPCONF/ISPF.conf
#FEL_UBLD_STEPLIB=$STEPLIB
```

*Figure 2. zee.env: environment configuration file*

The following definitions are optional. If omitted, default values are used.

**ELAXF**
The JES PROCLIB data set holding the ELAXF include member. The SET `variable=value` statements within will be converted to environment variables. There is no default. Change to match your z/OS Explorer Extensions customization.

**BZUINCL**
The JES PROCLIB data set holding the BZUINCL include member. The SET `variable=value` statements within will be converted to environment variables. There is no default. Change to match your z/OS Dynamic Test Runner customization.

**FEL_HLQ**
The high-level qualifier used to install z/OS Explorer Extensions. The default is FEL. Uncomment and change to match the location of your z/OS Explorer Extensions data sets.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDISABLE_DBM_INTEGRATION=false"**
Disable integration with Debug Manager, an optional started task of z/OS Debugger. The default is `false`, which implies that Debug Miner will attempt to connect at least once to Debug Manager. Uncomment and specify `true` to prevent any attempt to connect to Debug Manager.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Ddebug.miner.autoreconnect=0"**

Automatic reconnect to Debug Manager, an optional started task of z/OS Debugger. The default is 0, which implies that, when a connection with the Debug Manager server is not established or lost, the Debug Miner will attempt every minute to reconnect to the Debug Manager. Uncomment and specify a different value to limit how often the Debug Miner will attempt to connect to the Debug Manager.

*Table 11. Automatic reconnect to Debug Manager*

| **debug.miner.autoreconnect** | **Reconnect behavior** |
|---|---|
| -1 | Do not reconnect |
| 0 (default) | Attempt to reconnect every minute until successful |
| 1-86400 | Attempt to reconnect up to the specified amount of times. The maximum value, 86400, equals 24 hours. |

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Ddebug.miner.localhost=localhost"**
Alternative for the localhost TCP/IP definition. Debug Miner will attempt to connect to the Debug Manager, an optional started task of z/OS Debugger, using the localhost specification. This will fail if localhost does not resolve to the local loopback address (127.0.0.1 for IVPv4, ::1 for IPv6). Uncomment and specify the local loopback address when required.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dinclude.c=/etc/zexpl/include.conf"**
This variable points to a fully qualified z/OS UNIX file containing a list of forced includes for content assist on C code. A forced include consists of a file or directory, data set, or data set member which is parsed when a content assist operation is performed, regardless of whether that file or member was included in the source code using a pre-processor directive. To specify the name of the configuration file, uncomment and customize.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dinclude.cpp=/etc/zexpl/include.conf"**
This variable points to a fully qualified z/OS UNIX file containing a list of forced includes for content assist on C++ code. A forced include consists of a file or directory, data set, or data set member which is parsed when a content assist operation is performed, regardless of whether that file or member was included in the source code using a pre-processor directive. To specify the name of the configuration file, uncomment and customize.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DCPP_CLEANUP_INTERVAL=60000"**
Cleanup interval for unused C/C++ header files in milliseconds. The default is `60000`, which means 1 minute. To change the cleanup interval, Uncomment and customize. Specifying a value of 0 prevents caching of C/C++ header files, thereby reducing performance of remote content assist in the editor.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DRIS_BUFFER=8"**
Buffer size, in megabytes, used during remote index creation. The default is 8 MB. To change the buffer size, uncomment and customize. Valid values are whole numbers between 1 and 2000 (both inclusive). A bigger buffer speeds up index creation, but uses a bigger portion of the thread pool's Java heap. The buffer is automatically flushed to the index if it is full before index creation ends.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDISABLE_REMOTE_INDEX_SEARCH=true"**
Disable the Remote Index Search menu item on the client. The default is `false`. To prevent users from creating remote indexes for host system data sets, uncomment and specify `true`.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDEFAULT_REMOTE_ZOS_FILE_SEARCH=true"**
Make Remote z/OS File Search (using ISRSUPC) or Remote z/OS Search (using Java) the default panel on the client when starting a remote search. The default is false, which results in the "Remote z/OS

Search" panel being the default for the client. Uncomment and specify true to make Remote z/OS File Search the default panel on the client.

**Note:** This option requires the client to be v14.2.2 or later.

**_RSE_JAVAOPTS="$_RSE_JAVAOPTS -DDISABLE_DELETE_IN_SUBPROJECT=true"**
Disable the Delete menu item in the pop-up menu of z/OS subprojects. The default is `false`. To prevent users from using the Delete menu item in the pop-up menu of z/OS subprojects, uncomment and specify `true`.

**#_RSE_JAVAOPTS="$_RSE_JAVAOPTS -Dsclm.check.enable=true"**
Enable a check to see if a data set or member is under SCLM control. The default is false. Uncomment and specify true to prevent users from altering SCLM-managed data outside of SCLM.

**FEL_EDIT_MACRO_DSN**
Specifies the name of the data set that holds the REXX exec used for ISPF EDIT Macro support. When using defaults, FEL_EDIT_MACRO_DSN is set to FEL.SFELSAMP. Uncomment and change to match the location of a customized REXX exec, who's name is referenced in variable FEL_EDIT_MACRO.

**FEL_EDIT_MACRO**
Specifies the name of the REXX exec used for ISPF EDIT Macro support. The default is FELEDTMC. Uncomment and change to match the name of a customized REXX exec, who's location is referenced in variable FEL_EDIT_MACRO_DSN.

**STEPLIB**
Access MVS data sets not in LINKLIST/LPA. The default is `NONE`.

You can bypass the need of having prerequisite libraries in LINKLIST/LPA by uncommenting and customizing one or more of the following STEPLIB directives. For more information about the usage of the libraries in the following list, see “PARMLIB changes” on page 14:

```
# ZUnit, xUnit support for Enterprise COBOL and PL/I
STEPLIB=$STEPLIB:$FEL_HLQ.SFELLOAD:SYS1.CSSLIB:SYS1.SIXMLOD1
```

**Note:**

- Using STEPLIB in z/OS UNIX has a negative performance impact.
- If one STEPLIB library is APF-authorized, then all the other STEPLIB libraries must be authorized. Libraries lose their APF authorization when they are mixed with non-authorized libraries in STEPLIB.
- Libraries that are designed for LPA placement might require additional program control and APF authorizations if they are accessed through LINKLIST or STEPLIB.
- Coding a STEPLIB DD statement in the server JCL does not set the requested STEPLIB concatenation.

**AZU_JES_PROCLIB**
Define where cataloged procedures are stored so ZUnit can analyze them when adding recording parameters to JCL using a cataloged procedure. The default is SYS1.PROCLIB. Uncomment and specify a colon (:) separated list of data sets.

**FEL_UBLD_DD**
Specifies the DD statements that will be used when generating JCL for IBM Engineering Workflow Management (formerly known as IBM Rational Team Concert) user builds from a Developer for z/OS client that invoke TSO or ISPF commands. By default, Developer for z/OS uses the definitions in `ISPF.conf`, which is referenced by `CGI_ISPCONF` in `rse.env`. Uncomment and change to use the DD definitions in the specified file, which must follow the syntax rules specified for `ISPF.conf` in *IBM Explorer for z/OS Host Configuration Guide*.

**FEL_UBLD_STEPLIB**
Specifies the STEPLIB statement that will be used when generating JCL for IBM Engineering Workflow Management (formerly known as IBM Rational Team Concert) user builds from a Developer for z/OS client that invoke TSO or ISPF commands. By default, Developer for z/OS uses the `STEPLIB` definition in `rse.env`. Uncomment and change to use the specified `STEPLIB` definition.

## Specific components

z/OS Explorer Extensions consists of various unrelated features, each with their own customization tasks. Follow the instructions in the appropriate section to configure the required service.

Customizations to Developer for z/OS stand-alone components:

- Chapter 3, “Common Access Repository Manager (CARMA),” on page 27
- Chapter 4, “Host-based code analysis,” on page 57

Customizations to Developer for z/OS configuration files:

- “include.conf, Forced includes for C/C++ content assist” on page 61

Developer for z/OS related customizations to or for other products:

- “z/OS UNIX subprojects” on page 62
- “Include preprocessor support” on page 63
- “xUnit support for Enterprise COBOL and PL/I” on page 64
- “Enterprise Service Tools support” on page 69
- “CICS bidirectional language support” on page 69
- “Diagnostic IRZ messages for Enterprise Service Tools” on page 70
- “xUnit support for CICS applications (ZUnit)” on page 65

## Installation verification

The detailed description of the various installation verification programs (IVPs) is located in Chapter 6, “Installation verification,” on page 71.

# Chapter 3. Common Access Repository Manager (CARMA)

Common Access Repository Manager (CARMA) is a server platform for Repository Access Managers (RAMs). A RAM is an Application Programming Interface (API) for a z/OS based Software Configuration Manager (SCM). By wrapping the SCM functionality in a RAM, a single API is available for a client to access any supported SCM.

Developer for z/OS provides multiple pre-built RAMs and source code examples for creating your own RAM.

SCMs that are based on host systems need single-user address spaces to access their services, which requires CARMA to start a CARMA server for each user. It is not possible to create a single server supporting multiple users.

## Requirements and checklist

You need the assistance of a security administrator and a TCP/IP administrator to complete this customization task, which requires the following resources or special customization tasks:

- (Optional) TCP/IP port range for internal communication
- (Optional) Security rule to allow developers update capability to CARMA VSAM files
- (Optional) Security rule to allow users to submit CRA* jobs
- (Optional) LPA update

To start using CARMA at your site, do the following tasks. Unless otherwise indicated, all tasks are mandatory.

1. Choose a method to start CARMA and choose which RAMs should be activated. Several combinations of RAMs and server startup methods are available as a preconfigured setup. For details, see “Select the server startup method and active RAM” on page 28.
2. Create CARMA VSAM data sets. For details, see “CARMA VSAM data sets” on page 43 and “CARMA Repository Access Managers (RAMs)” on page 44.
3. Initial customization of the RSE configuration files to interface with CARMA. The complete customization is dependent on the method chosen to start CARMA. For details, see “CRASRV.properties, the RSE interface to CARMA” on page 36.
4. Depending on the chosen CARMA startup method and the chosen RAMs, do the required customization of the related configuration files. For details see:
   - “crastart*.conf, the CRASTART server startup” on page 39
   - “CRASUB*, the batch submit server startup” on page 42
5. Optionally, customize the CA Endevor® SCM-specific configuration members. For details see “CRACFG, CRASCL, CRASHOW and CRATMAP, the CA Endevor® SCM RAM configuration files” on page 46 and “CA Endevor® SCM RAM batch actions” on page 48.
6. Optionally, update the data set allocation exec. For details, see “CRANDVRA, the CA Endevor® SCM RAM initial allocation exec” on page 47, “CRAALLOC, the custom RAM allocation exec” on page 50, and “(Optional) Custom allocation exec” on page 52.
7. Optionally, create a startup user exit. For details, see (Optional) CARMA user exit.
8. Optionally, create CRAXJCL as replacement for IRXJCL. For details, see “(Optional) IRXJCL versus CRAXJCL” on page 55.

© Copyright IBM Corp. 2000, 2024

**Note:** The sample members referenced in this chapter are located in FEL.#CUST.* and /etc/zexpl, unless you specified a different location when you customized and submitted the FEL.SFELSAMP(FELSETUP) job. For more details, see "Customization setup" on page 13.

# Select the server startup method and active RAM

Developer for z/OS supports multiple methods to start a CARMA server. Developer for z/OS also provides multiple Repository Access Managers (RAMs), which can be divided into two groups, production RAMs and sample RAMs. This publication describes several possible combinations of RAMs and server startup methods. Each of the described configuration scenarios is available as a preconfigured setup.

## CARMA server startup

Developer for z/OS supports multiple methods to start a CARMA server. Each method has benefits and drawbacks.

### CRASTART

The "CRASTART" method starts the CARMA server as a subtask within RSE. This method provides a very flexible setup by using a separate configuration file that defines data set allocations and program invocations that are needed to start a CARMA server. This method provides the best performance and uses the fewest resources, but requires that the CRASTART module be located in LPA.

### Batch submit

The "batch submit" method starts the CARMA server by submitting a job. This is the default method that is used in the provided sample configuration files. The benefit of this method is that the CARMA logs are easily accessible in the job output. It also allows the use of custom server JCL for each developer, which is maintained by the developer himself. However, this method uses one JES initiator for each developer who starts a CARMA server.

## Production RAMs

Production type RAMs are fully functional, pre-built RAMs that can be used to access an SCM in a production environment.

### CA Endevor® SCM RAM

The IBM Developer for z/OS Interface for CA Endevor® Software Configuration Manager gives Developer for z/OS clients direct access to CA Endevor® SCM.

### CA Endevor® SCM packages RAM

The CA Endevor® SCM packages RAM gives Developer for z/OS clients direct access to CA Endevor® SCM packages.

## Sample RAMs

Sample RAMs are provided for the purpose of testing the configuration of your CARMA environment and as examples for developing your own RAMs. Source code is included.

**Attention:** Do not use the provided sample RAMs in a production environment.

### PDS RAM

The PDS RAM gives a data set list similar to **MVS Files -> My Data Sets** in the Remote Systems view.

### Skeleton RAM

The skeleton RAM gives a functional framework that can be used as starting point to develop your own RAM.

### SCLM RAM

The SCLM RAM gives a basic entry into SCLM, ISPF's Software Configuration Manager. The SCLM RAM is not enabled by default.

## Preconfigured RAM and server startup combinations

Several combinations of RAMs and server startup methods are available as a preconfigured setup. The listed scenarios need only minor customization to fit your environment.

- "CRASTART with CA Endevor® SCM RAM" on page 29
- "CRASTART with sample RAMs" on page 31
- "Batch submit with CA Endevor® SCM RAM" on page 32
- "Batch submit with sample RAMs" on page 35

Detailed information on the different steps of each scenario can be found in "CARMA configuration details" on page 36.

It is possible to add a RAM to any CARMA setup, now or somewhere in the future. See "(Optional) Supporting multiple RAMs" on page 51 for more information on adding a RAM to an existing setup.

# CRASTART with CA Endevor® SCM RAM

The information in this section describes how to set up CARMA with the following specifications:

- Server startup: CRASTART method. This method requires that CRASTART is in LPA.
- RAM: CA Endevor® SCM RAM.

This customization step can be omitted if you want to use one of the other scenarios with different specifications.

## Create the CARMA VSAM data sets

To define and populate the CARMA-related VSAM data sets, customize and submit the following JCL jobs. For customization instructions, see the documentation within the member. Existing VSAM data sets are replaced.

For more details on this step, see "CARMA VSAM data sets" on page 43.

- `FEL.#CUST.JCL(CRA$VCAD)`
- `FEL.#CUST.JCL(CRA$VCAS)`
- `FEL.#CUST.JCL(CRA$VMSG)`

## Customize CRASRV.properties

RSE server uses the settings in `/etc/zexpl/CRASRV.properties` to start and connect to a CARMA server. You can edit the file with the TSO **`OEDIT`** command. For the changes to take effect, restart the RSED started task.

When you use the default file locations, the only required changes are changing the value of the `clist.dsname` directive to `*CRASTART` and changing the value of `crastart.configuration.file` to `crastart.endevor.conf`. For more information about the different directives, see "CRASRV.properties, the RSE interface to CARMA" on page 36.

```
clist.dsname=*CRASTART
crastart.configuration.file=crastart.endevor.conf
```

*Figure 3. CRASRV.properties: CRASTART with CA Endevor® SCM RAM*

## Customize crastart.endevor.conf

CRASTART uses the definitions in `/etc/zexpl/crastart.endevor.conf` to create a valid TSO/ISPF environment to start a CARMA server. You can edit the file with the TSO **OEDIT** command. Changes are in effect for all CARMA servers that are started after the update.

For customization instructions, see the documentation within the file. For more information about the CRASTART startup method, see "crastart*.conf, the CRASTART server startup" on page 39.

**Note:** Due to page width limitations, some lines in the following sample wrapped onto the next line. All lines that start with an indentation should be added to the end of the previous line.

```
* DD used by RAM
TYPEMAP = FEL.#CUST.PARMLIB(CRATMAP)
SHOWVIEW= FEL.#CUST.PARMLIB(CRASHOW)
CRACFG  = FEL.#CUST.PARMLIB(CRACFG)
* uncomment CRABCFG and CRABSKEL to use batch actions
*CRABCFG = FEL.#CUST.PARMLIB(CRABCFG)
*CRABSKEL= FEL.#CUST.CNTL
* uncomment and provide correct DSN to use Package Ship
*APIHJC  = #shiphjc
CONLIB  = CA.NDVR.CSIQLOAD
-COMMAND=ALLOC FI(JCLOUT)   SYSOUT(A) WRITER(INTRDR) RECFM(F) LRECL(80)
  BLKSIZE(80)
-COMMAND=ALLOC FI(EXT1ELM)  NEW DELETE DSORG(PS) RECFM(V,B) LRECL(4096)
  BLKSIZE(27998) SPACE(5,5) TRACKS UNIT(SYSALLDA)
-COMMAND=ALLOC FI(EXT2ELM)  NEW DELETE DSORG(PS) RECFM(V,B) LRECL(4096)
  BLKSIZE(27998) SPACE(5,5) TRACKS UNIT(SYSALLDA)
-COMMAND=ALLOC FI(EXT1DEP)  NEW DELETE DSORG(PS) RECFM(V,B) LRECL(4096)
  BLKSIZE(27998) SPACE(5,5) TRACKS UNIT(SYSALLDA)
C1EXMSGS= SYSOUT(H)
C1MSGS1 = SYSOUT(H)
MSG3FILE= DUMMY

* DD used by CARMA server (CRASERV)
* pay attention to APF authorizations when using TASKLIB
TASKLIB = FEL.SFELLOAD,CA.NDVR.CSIQAUTH,CA.NDVR.CSIQAUTU
CRADEF  = FEL.#CUST.CRADEF
CRAMSG  = FEL.#CUST.CRAMSG
CRASTRS = FEL.#CUST.CRASTRS
CARMALOG= SYSOUT(H)
SYSPRINT= SYSOUT(H)

* DD used by ISPF (via NDVRC1)
-COMMAND=ALLOC FI(ISPCTL0) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA)
-COMMAND=ALLOC FI(ISPCTL1) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA)
-COMMAND=ALLOC FI(ISPPROF) NEW DELETE DSORG(PO) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA) DIR(5)
ISPTABL = -ISPPROF
ISPTLIB = -ISPPROF,ISP.SISPTENU
ISPMLIB = ISP.SISPMENU
ISPPLIB = ISP.SISPPENU
ISPSLIB = ISP.SISPSENU

* DD used by TSO (IKJEFT01)
SYSPROC = FEL.SFELPROC                                    * CRANDVRA
SYSTSIN = DUMMY
SYSTSPRT= SYSOUT(H)

PROGRAM=IKJEFT01 %CRANDVRA NDVRC1 PGM(CRASERV) PARM(&CRAPRM1.
   &CRAPRM2. &CRAPRM3. &CRAPRM4. &CRAPRM5. &CRAPRM6. &CRAPRM7.
   &CRAPRM8. )
```

*Figure 4. crastart.endevor.conf: CRASTART with CA Endevor® SCM RAM*

## (Optional) Additional CA Endevor® SCM RAM customization

The CA Endevor® SCM RAM has additional components that can be customized if needed.

- CARMA startup processing has an optional user exit. For more information see (Optional) CARMA user exit.
- The CA Endevor® SCM RAM has multiple configuration files FEL.#CUST.PARMLIB(CRA*) that can be customized. For more information, see "CRACFG, CRASCL, CRASHOW and CRATMAP, the CA Endevor® SCM RAM configuration files" on page 46.
- The CA Endevor® SCM RAM has an allocation exec, FEL.SFELPROC(CRANDVRA), that can be customized. For more information, see "CRANDVRA, the CA Endevor® SCM RAM initial allocation exec" on page 47.
- The CA Endevor® SCM RAM supports doing CA Endevor® SCM actions in batch mode. Batch-actions requires a configuration file, FEL.#CUST.PARMLIB(CRABCFG), and a skeleton JCL, FEL.#CUST.CNTL(CRABATCA), that must be customized. For more information, see "CA Endevor® SCM RAM batch actions" on page 48.

# CRASTART with sample RAMs

The information in this section describes how to set up CARMA with the following specifications:

- Server startup: CRASTART method. This method requires that CRASTART is in LPA.
- RAM: sample RAMs, which are not to be used for production purposes.

This customization step can be bypassed if you want to use one of the other scenarios with different specifications.

## Create the CARMA VSAM data sets

Customize and submit the following JCL jobs to define and populate the CARMA-related VSAM data sets. For customization instructions, see the documentation within the member. Existing VSAM data sets are replaced.

For more details on this step, see "CARMA VSAM data sets" on page 43 and "CARMA Repository Access Managers (RAMs)" on page 44.

### CARMA

- FEL.#CUST.JCL(CRA$VDEF)
- FEL.#CUST.JCL(CRA$VMSG)
- FEL.#CUST.JCL(CRA$VSTR)

### Sample RAMs

- FEL.#CUST.JCL(CRA#VPDS)

## Customize CRASRV.properties

RSE server uses the settings in /etc/zexpl/CRASRV.properties to start and connect to a CARMA server. You can edit the file with the TSO **OEDIT** command. For the changes to take effect, the RSED started task must be restarted.

When using the default file locations, the only required change is changing the value of the clist.dsname directive to *CRASTART. For more information about the different directives, see "CRASRV.properties, the RSE interface to CARMA" on page 36.

```
clist.dsname=*CRASTART
crastart.configuration.file=crastart.conf
```

*Figure 5. CRASRV.properties: CRASTART with sample RAMs*

## Customize crastart.conf

CRASTART uses the definitions in `/etc/zexpl/crastart.conf` to create a valid TSO/ISPF environment to start a CARMA server. You can edit the file with the TSO **OEDIT** command. Changes are in effect for all CARMA servers that are started after the update.

For customization instructions, see the documentation within the file. For more information about the CRASTART startup method, see "crastart*.conf, the CRASTART server startup" on page 39.

```
* DD used by RAM
CRARAM1 = FEL.#CUST.CRARAM1                                  * PDS RAM
* DD used by CARMA server (CRASERV)
TASKLIB = FEL.SFELLOAD
CRADEF  = FEL.#CUST.CRADEF
CRAMSG  = FEL.#CUST.CRAMSG
CRASTRS = FEL.#CUST.CRASTRS
CARMALOG= SYSOUT(H)
SYSPRINT= SYSOUT(H)

* DD used by ISPF (ISPSTART)
-COMMAND=ALLOC FI(ISPCTL0) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA)
-COMMAND=ALLOC FI(ISPCTL1) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA)
-COMMAND=ALLOC FI(ISPPROF) NEW DELETE DSORG(PO) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA) DIR(5)
ISPTABL = -ISPPROF
ISPTLIB = -ISPPROF,ISP.SISPTENU
ISPMLIB = ISP.SISPMENU
ISPPLIB = ISP.SISPPENU
ISPSLIB = ISP.SISPSENU

* DD used by TSO (IKJEFT01)
SYSPROC = #hlq.SFELPROC                                     * CRAALLOC
SYSTSIN = DUMMY
SYSTSPRT= SYSOUT(H)

PROGRAM=IKJEFT01 %CRAALLOC ISPSTART PGM(CRASERV) PARM(&CRAPRM1.
   &CRAPRM2. &CRAPRM3. &CRAPRM4. &CRAPRM5. &CRAPRM6. &CRAPRM7.
   &CRAPRM8. )
```

*Figure 6. crastart.conf: CRASTART with sample RAMs*

**Note:** Due to page width limitations, some lines in the sample wrapped onto the next line. All lines that start with an indentation should be added to the end of the previous line.

## (Optional) Additional custom RAM customization

The custom RAMs have additional components that can be customized if needed.

- CARMA startup processing has an optional user exit. For more information see (Optional) CARMA user exit.
- Custom RAM startup has an allocation exec, `FEL.SFELPROC(CRAALLOC)`, that can be customized. For more information, see CRAALLOC, the custom RAM allocation exec.

# Batch submit with CA Endevor® SCM RAM

The information in this section describes how to set up CARMA with the following specifications:

- Server startup: batch submit method. This method requires JES initiators.
- RAM: CA Endevor® SCM RAM.

This customization step can be omitted if you want to use one of the other scenarios with different specifications.

## Create the CARMA VSAM data sets

Customize and submit the following JCLs to define and populate the CARMA-related VSAM data sets. For customization instructions, see the documentation within the member. Existing VSAM data sets are replaced.

For more details on this step, see “CARMA VSAM data sets” on page 43.

- `FEL.#CUST.JCL(CRA$VCAD)`
- `FEL.#CUST.JCL(CRA$VCAS)`
- `FEL.#CUST.JCL(CRA$VMSG)`

## Customize CRASRV.properties

RSE server uses the settings in `/etc/zexpl/CRASRV.properties` to start and connect to a CARMA server. You can edit the file with the TSO **OEDIT** command. For the changes to take effect, the RSED started task must be restarted.

When using default file locations, the only required change is changing the value of the `clist.dsname` directive to `FEL.#CUST.CNTL(CRASUBCA)`. For more information about the different directives, see “CRASRV.properties, the RSE interface to CARMA” on page 36.

```
clist.dsname='FEL.#CUST.CNTL(CRASUBCA)'
```

*Figure 7. CRASRV.properties: Batch submit with CA Endevor® SCM RAM*

## Customize CRASUBCA

The `FEL.#CUST.CNTL(CRASUBCA)` CLIST and embedded JCL submits a CARMA server. Changes are in effect for all CARMA servers that are started after the update.

For customization instructions, see the documentation within the member. For more information about the batch submit startup method, see “CRASUB*, the batch submit server startup” on page 42.

```
PROC 8 CRAPRM1 CRAPRM2 CRAPRM3 CRAPRM4 CRAPRM5 CRAPRM6 CRAPRM7 CRAPRM8
SUBMIT * END($$)
//CRA&PORT JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//*
//RUN      EXEC PGM=IKJEFT01,DYNAMNBR=125,REGION=0M,TIME=NOLIMIT
//*
//* DD used by RAM
//TYPEMAP  DD DISP=SHR,DSN=FEL.#CUST.PARMLIB(CRATMAP)
//SHOWVIEW DD DISP=SHR,DSN=FEL.#CUST.PARMLIB(CRASHOW)
//CRACFG   DD DISP=SHR,DSN=FEL.#CUST.PARMLIB(CRACFG)
//* uncomment CRABCFG and CRABSKEL to use batch actions
//*CRABCFG  DD DISP=SHR,DSN=FEL.#CUST.PARMLIB(CRABCFG)
//*CRABSKEL DD DISP=SHR,DSN=FEL.#CUST.CNTL
//* uncomment and provide correct DSN to use Package Ship
//*APIHJC  DD DISP=SHR,DSN=#shiphjc
//CONLIB   DD DISP=SHR,DSN=CA.NDVR.CSIQLOAD
//JCLOUT   DD SYSOUT=(A,INTRDR),DCB=(LRECL=80,RECFM=F,BLKSIZE=80)
//EXT1ELM  DD DISP=(NEW,DELETE),UNIT=SYSALLDA,
//            RECFM=VB,LRECL=4096,BLKSIZE=27998,SPACE=(TRK,(5,5))
//EXT2ELM  DD DISP=(NEW,DELETE),UNIT=SYSALLDA,
//            RECFM=VB,LRECL=4096,BLKSIZE=27998,SPACE=(TRK,(5,5))
//EXT1DEP  DD DISP=(NEW,DELETE),UNIT=SYSALLDA,
//            RECFM=VB,LRECL=4096,BLKSIZE=27998,SPACE=(TRK,(5,5))
//C1MSGS1  DD SYSOUT(H)
//C1EXMSGS DD SYSOUT(H)
//MSG3FILE DD DUMMY
//*
//* DD used by CARMA server (CRASERV)
//* pay attention to APF authorizations when using STEPLIB
//STEPLIB  DD DISP=SHR,DSN=FEL.SFELLOAD
//         DD DISP=SHR,DSN=CA.NDVR.CSIQAUTH
//         DD DISP=SHR,DSN=CA.NDVR.CSIQAUTU
//CRADEF   DD DISP=SHR,DSN=FEL.#CUST.CRADEF
//CRAMSG   DD DISP=SHR,DSN=FEL.#CUST.CRAMSG
//CRASTRS  DD DISP=SHR,DSN=FEL.#CUST.CRASTRS
//CARMALOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*
//* DD used by ISPF (via NDVRC1)
//ISPPROF  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(1,1,5))
//ISPCTL0  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(5,5))
//ISPCTL1  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(5,5))
//ISPMLIB  DD DISP=SHR,DSN=ISP.SISPMENU
//ISPPLIB  DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB  DD DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB  DD DISP=SHR,DSN=ISP.SISPTENU
//*
//* DD used by TSO (IKJEFT01)
//SYSPROC  DD DISP=SHR,DSN=FEL.SFELPROC                        * CRANDVRA
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
%CRANDVRA NDVRC1 PGM(CRASERV) PARM(&CRAPRM1 &CRAPRM2 &STR(+)
&CRAPRM3 &STR(+)
&CRAPRM4 &STR(+)
&CRAPRM5 &STR(+)
&CRAPRM6 &STR(+)
&CRAPRM7 &STR(+)
&CRAPRM8 &STR(+) )
$$
EXIT CODE(0)
```

*Figure 8. CRASUBCA: Batch submit with CA Endevor® SCM RAM*

## (Optional) Additional CA Endevor® SCM RAM customization

The CA Endevor® SCM RAM has additional components that can be customized if needed.

- CARMA startup processing has an optional user exit. For more information see "(Optional) CARMA user exit" on page 53.
- The CA Endevor® SCM RAM has multiple configuration files, `FEL.#CUST.PARMLIB(CRACFG)`, `FEL.#CUST.PARMLIB(CRASHOW)` and `FEL.#CUST.PARMLIB(CRATMAP)`, that can be customized. For more information, see "CRACFG, CRASCL, CRASHOW and CRATMAP, the CA Endevor® SCM RAM configuration files" on page 46.

- The CA Endevor® SCM RAM has an allocation exec, FEL.SFELPROC(CRANDVRA), that can be customized. For more information, see "CRANDVRA, the CA Endevor® SCM RAM initial allocation exec" on page 47.
- The CA Endevor® SCM RAM supports doing CA Endevor® SCM actions in batch mode. Batch-actions requires a configuration file, FEL.#CUST.PARMLIB(CRABCFG), a skeleton JCL, FEL.#CUST.CNTL(CRABATCA), and an optional default job card, FEL.#CUST.CNTL(CRABJOBC), that must be customized. For more information, see "CA Endevor® SCM RAM batch actions" on page 48.

# Batch submit with sample RAMs

The information in this section describes how to set up CARMA with the following specifications:

- Server startup: batch submit method, which requires JES initiators
- RAM: sample RAMs, which are not to be used for production purposes

This customization step can be omitted if you want to use one of the other scenarios with different specifications.

## Create the VSAM data sets

Customize and submit the following JCL jobs to define and populate the CARMA-related VSAM data sets. For customization instructions, see the documentation within the member. Existing VSAM data sets are replaced.

For more details on this step, see "CARMA VSAM data sets" on page 43 and "CARMA Repository Access Managers (RAMs)" on page 44.

### CARMA

- FEL.#CUST.JCL(CRA$VDEF)
- FEL.#CUST.JCL(CRA$VMSG)
- FEL.#CUST.JCL(CRA$VSTR)

### Sample RAMs

- FEL.#CUST.JCL(CRA#VPDS)

## Customize CRASRV.properties

RSE server uses the settings in /etc/zexpl/CRASRV.properties to start and connect to a CARMA server. You can edit the file with the TSO **OEDIT** command. For the changes to take effect, the RSED started task must be restarted.

When using default file locations, the only required change is changing the value of the clist.dsname directive to FEL.#CUST.CNTL(CRASUBMT). For more information about the different directives, see "CRASRV.properties, the RSE interface to CARMA" on page 36.

```
clist.dsname='FEL.#CUST.CNTL(CRASUBMT)'
```

*Figure 9. CRASRV.properties: Batch submit with sample RAMs*

## Customize CRASUBMT

The FEL.#CUST.CNTL(CRASUBMT) CLIST and embedded JCL submits a CARMA server. Changes are in effect for all CARMA servers that are started after the update.

For customization instructions, see the documentation within the member. For more information about the batch submit startup method, see "CRASUB*, the batch submit server startup" on page 42.

```
PROC 8 CRAPRM1 CRAPRM2 CRAPRM3 CRAPRM4 CRAPRM5 CRAPRM6 CRAPRM7 CRAPRM8
SUBMIT * END($$)
//CRA&PORT JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//*
//RUN      EXEC PGM=IKJEFT01,DYNAMNBR=125,REGION=0M,TIME=NOLIMIT
//*
//* DD used by RAM
//CRARAM1  DD DISP=SHR,DSN=FEL.#CUST.CRARAM1            * PDS RAM
//*
//* DD used by CARMA server (CRASERV)
//STEPLIB  DD DISP=SHR,DSN=FEL.SFELLOAD
//CRADEF   DD DISP=SHR,DSN=FEL.#CUST.CRADEF
//CRAMSG   DD DISP=SHR,DSN=FEL.#CUST.CRAMSG
//CRASTRS  DD DISP=SHR,DSN=FEL.#CUST.CRASTRS
//CARMALOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*
//* DD used by ISPF (ISPSTART)
//ISPPROF  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(1,1,5))
//ISPCTL0  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(5,5))
//ISPCTL1  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(5,5))
//ISPMLIB  DD DISP=SHR,DSN=ISP.SISPMENU
//ISPPLIB  DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB  DD DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB  DD DISP=SHR,DSN=ISP.SISPTENU
//*
//* DD used by TSO (IKJEFT01)
//SYSPROC  DD DISP=SHR,DSN=#hlq.SFELPROC                * CRAALLOC
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
%CRAALLOC ISPSTART PGM(CRASERV) PARM(&CRAPRM1 &CRAPRM2 &STR(+)
&CRAPRM3 &STR(+)
&CRAPRM4 &STR(+)
&CRAPRM5 &STR(+)
&CRAPRM6 &STR(+)
&CRAPRM7 &STR(+)
&CRAPRM8 &STR(+)
)
$$
EXIT CODE(0)
```

*Figure 10. CRASUBMT: Batch submit with sample RAMs*

## (Optional) Additional custom RAM customization

The custom RAMs have additional components that can be customized if needed.

- CARMA startup processing has an optional user exit. For more information see (Optional) CARMA user exit.
- Custom RAM startup has an allocation exec, FEL.SFELPROC(CRAALLOC), that can be customized. For more information, see CRAALLOC, the custom RAM allocation exec.

# CARMA configuration details

The different configuration scenarios that are documented in this publication share many of the CARMA configuration files. The details of these configuration files are documented here, and they are referenced from within the various scenarios.

## CRASRV.properties, the RSE interface to CARMA

The Common Access Repository Manager (CARMA) server provides a standard API for other products that use host systems to access one or more Software Configuration Managers (SCMs). However, it does not provide methods for direct communication with a client computer. For this communication, it relies on other products, such as the Remote System Explorer (RSE) server. The RSE server uses the settings in CRASRV.properties to start and connect to a CARMA server.

CRASRV.properties is located in /etc/zexpl/, unless you specified a different location when customizing and submitting the FEL.SFELSAMP(FELSETUP) job. For more details, see "Customization setup" on page 13. You can edit the file with the TSO **OEDIT** command.

**Note:** For the changes to take effect, the Remote System Explorer Daemon (RSED) started task must be restarted.

```
# CRASRV.properties - CARMA configuration options
#
clist.dsname=''
crastart.configuration.file=crastart.conf
#connect.timeout=60
#port.start=0
#port.range=100
#system.exit='/usr/lpp/IBM/zee/samples/craexit.rex'
#user.exit='FEL.SFELSAMP(CRAEXIT)'
#startup.script.name=carma.startup.rex
#crastart.stub=CRASTART
#crastart.syslog=Partial
#crastart.timeout=420
#crastart.steplib=FEL.SFELLPA
#crastart.tasklib=TASKLIB
```

*Figure 11. CRASRV.properties – CARMA configuration file*

**clist.dsname**
Defines the startup method for the CARMA server. For more details about the different startup methods, see "Select the server startup method and active RAM" on page 28.

- *CRASTART indicates that the CARMA server should be started as a subtask within RSE using CRASTART. If you specify *CRASTART, you must also specify the crastart.* directives, or use their default values.
- Any other value defines the location of the CRASUBMT CLIST, using TSO-like naming conventions. With single quotation marks (') the data set name is an absolute reference, without the single quotation marks (') the data set name is prefixed with the client's user ID, not the TSO prefix. The latter requires that all CARMA users must maintain their own CRASUBMT CLIST.

The default is a null string, to indicate that CARMA is not configured.

**crastart.configuration.file**
Specifies the name of the CRASTART configuration file. The default is crastart.conf. This file specifies the data set allocations and program invocations that are needed to start a CARMA server. This directive is used only if the clist.dsname directive has *CRASTART as value. The file name can be specified in several ways:

- Null string, which means that the variable is not specified. The default value is used.
- Only a file name, which is the default method. CARMA searches your configuration directory (/etc/zexpl by default) to find the file.
- Relative path, which is the directory and file name, without a leading forward slash (/). CARMA adds your configuration directory (/etc/zexpl/ by default) to the provided path to make it an absolute path.
- Absolute path, which is the directory and file name, with a leading forward slash (/). CARMA uses the specified file location.

**#connect.timeout**
Specifies (in seconds) how long CARMA miner (active in RSE) waits for CARMA server (load module CRASERV) to start up and connect to the port on which CARMA miner is listening. The default is 60 seconds. Uncomment and customize to change the wait time.

**#port.start**
When the value of port.start is 0 (zero), CARMA uses an ephemeral port for communication between CARMA and the RSE server. In this scenario, TCP/IP assigns a random free port number. When the value of port.start is non-zero, it is interpreted as the starting point of a port range used for communication between CARMA and the RSE server, in which case the port.range variable

must also be defined. The default port is `0`. To specify the start of the port range, uncomment and customize. Communication on this port is confined to your host system.

**Note:** Before selecting a port, verify that the port is available on your system by using the **NETSTAT** and **NETSTAT PORTL** commands. For more information, see "Reserved TCP/IP ports" in the *Host Configuration Reference (SC27-9934)* .

**#port.range**
Range of ports, starting at `port.start`, which is used for CARMA communication if `port.start` is non-zero. The default is `100`. To specify the size of the port range, uncomment and customize. For example, when `port.start` is 5227 and `port.range` is 100, port 5227 until 5326 (both inclusive) can be used by CARMA. Each CARMA connection uses a port exclusively, so specifying a port range limits the maximum number of concurrent CARMA sessions.

**#system.exit**
Defines user-specified code to be executed before creation of the CARMA server address space. The user-specified code is also executed after termination of the CARMA server address space to allow for cleanup actions. Uncomment and specify the (z/OS UNIX) file name of the code to be executed. The file name can be specified in several ways:

- Only a file name. CARMA searches the directories in the PATH environment variable to find the file.
- Relative path, which is the directory and file name, without a leading forward slash (/). CARMA adds your configuration directory (`/etc/zexpl/` by default) to the provided path to make it an absolute path.
- Absolute path, which is the directory and file name, with a leading forward slash (/). CARMA uses the specified file location.

A sample user exit is provided as `/usr/lpp/IBM/zee/samples/craexit.rex`. This sample also documents the startup arguments passed to the user exit. For more information see (Optional) CARMA user exit.

**#user.exit**
Defines user-specified code to be executed after the creation of the CARMA server address space, just before the CARMA server is started. The user-specified code is also executed after the CARMA server ends, before address space termination, to allow for cleanup actions.. Uncomment and specify the data set name of the code to be executed.

With quotes (') the data set name is an absolute reference, without quotes (') the data set name is prefixed with the client's user ID, not the TSO prefix. The latter requires that all CARMA users must maintain their own exit code.

A sample user exit is provided as `FEL.SFELSAMP(CRAEXIT)`. This sample also documents the startup arguments passed to the user exit. For more information see (Optional) CARMA user exit.

**startup.script.name**
Defines the CARMA startup script. The default is `carma.startup.rex`. This REXX exec triggers the startup of a CARMA server. The file name can be specified in several ways:

- Null string, which means that the variable is not specified. In this case, the default value is used.
- Only a file name, which is the default method. CARMA searches the directories in the PATH environment variable to find the file. The directory holding Developer for z/OS executables (`/usr/lpp/IBM/zee/bin` by default) is automatically added to the PATH environment variable.
- Relative path, which is the directory and file name, without a leading forward slash (/). CARMA adds your configuration directory (`/etc/zexpl/` by default) to the provided path to make it an absolute path.
- Absolute path, which is the directory and file name, with a leading forward slash (/). CARMA uses the specified file location.

**#crastart.stub**
z/OS UNIX stub for calling CRASTART. The default is CRASTART. This stub makes the MVS based CRASTART load module available to z/OS UNIX processes. To specify a specific path, uncomment and

customize. This directive is used only if the `clist.dsname` directive has `*CRASTART` as value. The file name can be specified in several ways:

- Null string, which means that the variable is not specified. The default value is used.
- Only a file name, which is the default method. CARMA searches the directories in the PATH environment variable to find the file. The directory holding Developer for z/OS executables (`/usr/lpp/IBM/zee/bin` by default) is automatically added to the PATH environment variable.
- Relative path, which is the directory and file name, without a leading forward slash (/). CARMA adds your configuration directory (`/etc/zexpl/` by default) to the provided path to make it an absolute path.
- Absolute path, which is the directory and file name, with a leading forward slash (/). CARMA uses the specified file location.

**#crastart.syslog**
Specifies how much information is written to the system log while CRASTART starts a CARMA server. The default is `Partial`. Valid values are listed in the following table.

| Value | Description |
|---|---|
| A (All) | All tracing information is printed to SYSLOG |
| P (Partial) | Only connect, disconnect, and error information is printed to SYSLOG |
| anything else | Only error conditions are printed to SYSLOG |

To specify the required detail level for system log messages, uncomment and customize. This directive is used only if the `clist.dsname` directive has `*CRASTART` as value.

**#crastart.timeout**
The length of time, in seconds, before a CARMA server ends due to lack of activity. The default is `420` (7 minutes). To specify the required timeout value, uncomment and customize. This directive is used only if the `clist.dsname` directive has `*CRASTART` as value.

**Note:** System abend 522 for module CRASERV will occur if the JWT parameter in the SMFPRMxx parmlib member is set to a value lower than the `crastart.timeout` value in `CRASRV.properties`. This occurrence does not impact CARMA operations because the server is restarted automatically if needed.

**#crastart.steplib**
The location of the CRASTART module when accessed through the STEPLIB directive in `zee.env`. The default is `FEL.SFELLPA`. If the CRASTART module cannot be part of LPA or LINKLIST, uncomment and customize this directive. Program control and APF issues might arise if the CRASTART module is not in LPA. This directive is used only if the `clist.dsname` directive has `*CRASTART` as value.

**#crastart.tasklib**
Alternate name for the TASKLIB DD name in `crastart.conf`. The default is TASKLIB. If the DD name TASKLIB has a special meaning for your SCM or RAM and cannot be used as STEPLIB replacement, uncomment and customize this directive. This directive is used only if the `clist.dsname` directive has `*CRASTART` as value.

## crastart*.conf, the CRASTART server startup

RSE starts the CRASTART load module, which uses the definitions in `crastart*.conf` to create a valid environment to execute batch TSO and ISPF commands. Developer for z/OS uses this environment to run the CARMA server, CRASERV.

`crastart*.conf` is located in `/etc/zexpl/`, unless you specified a different location when you customized and submitted job `FEL.SFELSAMP(FELSETUP)`. For more details, see "Customization setup" on page 13. You can edit the file with the TSO **OEDIT** command.

**Note:** Changes are in effect for all CARMA servers that are started after the update.

Developer for z/OS provides multiple `crastart*.conf` configuration files. Each of these sample files is preconfigured for a specific customization scenario:

- `crastart.endevor.conf` is configured for CRASTART startup with CA Endevor® SCM RAM.
- `crastart.conf` is configured for CRASTART startup with sample RAMs.

The function of the `crastart*.conf` file is similar in concept to a JCL job stream, but is more restrictive.

- The following samples show valid line formats:
  - `* comment`
  - `ddname=dsn1,dsn2,dsn3 * comment`
  - `ddname=SYSOUT(c) * comment`
  - `ddname=DUMMY * comment`
  - `-COMMAND=<any bpxwdyn command> * comment`
  - `PROGRAM = progname parms * comment`

  **Note:** The **BPXWDYN** command is documented in *Using REXX and z/OS UNIX System Services* (SA22-7806) and allows complex allocation constructs.
- All input is changed to uppercase.
- Line continuations are not supported.
- There is no limitation on line length.
- One or more blank spaces are allowed around the equal sign (=).
- DD allocations must precede the related PROGRAM statement.
- DD names allocated here are freed at the end of program execution. They do not accumulate.
- DD names allocated by the called programs are not freed.
- Multiple data sets can be concatenated to a DD name. The data set names must be separated by a comma (,), and the concatenation is searched in the listed order.
- All data set allocations are done with DISP=SHR, except for allocations done using `-COMMAND`.
- Inline data is not supported. All data must be in cataloged files.
- Variables can be used only on the right side of the equal sign (=).
- The following variables are supported:

| Variable | Description |
|---|---|
| &CRAUSER. | Client user ID |
| &CRADATE. | Current® date in `Dyyyyddd` format (7 char Julian) |
| &CRATIME. | Current time in `Thhmmss` format (hour min sec) |
| &CRAPRM1. | Port number |
| &CRAPRM8. | Site-specific value set by the user exit referenced by the `system.exit` directive in `CRASRV.properties`. |
| System symbol | Any `SYS1.PARMLIB(IEASYMxx)` system symbol |
| -<ddname> | A hyphen (-) followed by a previously defined DD name acts like a `*.ddname` backward reference in JCL. The original DD must be allocated using the `–COMMAND` statement. |

**Note:** There is no variable for the TSO prefix because TSO is not active when the configuration file is interpreted. If you have a need for the TSO prefix or other variable that is not available, see "(Optional) Custom allocation exec" on page 52.

Figure 12 on page 41 shows a basic `crastart*.conf` skeleton that includes ISPF services.

```
* DD used by RAM

* DD used by CARMA server (CRASERV)
TASKLIB = FEL.SFELLOAD
CRADEF  = FEL.#CUST.CRADEF
CRAMSG  = FEL.#CUST.CRAMSG
CRASTRS = FEL.#CUST.CRASTRS
CARMALOG= SYSOUT(H)
SYSPRINT= SYSOUT(H)

* DD used by ISPF (ISPSTART)
-COMMAND=ALLOC FI(ISPCTL0) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA)
-COMMAND=ALLOC FI(ISPCTL1) NEW DELETE DSORG(PS) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA)
-COMMAND=ALLOC FI(ISPPROF) NEW DELETE DSORG(PO) RECFM(F,B) LRECL(80)
  BLKSIZE(32720) SPACE(5,5) TRACKS UNIT(SYSALLDA) DIR(5)
ISPTABL = -ISPPROF
ISPTLIB = -ISPPROF,ISP.SISPTENU
ISPMLIB = ISP.SISPMENU
ISPPLIB = ISP.SISPPENU
ISPSLIB = ISP.SISPSENU

* DD used by TSO (IKJEFT01)
SYSPROC = #hlq.SFELPROC                                  * CRAALLOC
SYSTSIN = DUMMY
SYSTSPRT= SYSOUT(H)

PROGRAM=IKJEFT01 %CRAALLOC ISPSTART PGM(CRASERV) PARM(&CRAPRM1.
   &CRAPRM2. &CRAPRM3. &CRAPRM4. &CRAPRM5. &CRAPRM6. &CRAPRM7.
   &CRAPRM8. )
```

*Figure 12. crastart*.conf: CARMA server startup using CRASTART*

**Note:**

- Due to page width limitations, some lines in the sample wrapped onto the next line. All lines that start with an indentation should be added to the end of the previous line.
- If you alter the PROGRAM line, ensure that there is at least one blank before the closing round bracket (“)”) of the PARM() statement to simplify processing of the string.
- You can add your own DD statements and data set concatenations to customize the CARMA TSO environment, thus mimicking a TSO logon procedure.
- The DD name TASKLIB acts like STEPLIB in JCL. Its DD name must match the value specified for `crastart.tasklib` in CRASRV.`properties`, which is described in “CRASRV.properties, the RSE interface to CARMA” on page 36.
- Regular APF rules apply for TASKLIB allocations. Libraries lose their APF authorization when a non-APF authorized library is part of the concatenation.
- System abend 522 for module CRASERV occurs if the JWT parameter in the SMFPRMxx parmlib member is set to a value lower than the `crastart.timeout` value in CRASRV.`properties`. The system abend does not impact CARMA operations because the server is restarted automatically if needed.
- Details of the CARMA server startup are shown in `rsecomm.log` when the server ends. For more information on setting the detail level of `rsecomm.log`, see the *IBM Explorer for z/OS Host Configuration Guide (SC27-8437)*.

## Collecting the CRASTART log files

CRASTART creates a TSO environment as a child process of RSE, which runs in a separate address space. Non-trivial actions might be needed to keep the CARMA output sent to SYSOUT(*), which complicates the collecting of log files. This difficulty can be resolved by writing the log files to a user-specific data set, as shown in the following sample allocation:

```
-COMMAND=ALLOC FI(CARMALOG) MOD CATALOG DSORG(PS) RECFM(F,B) LRECL(133)
  BLKSIZE(27930) SPACE(5,5) TRACKS UNIT(SYSALLDA)
  DA(&CRAUSER..&SYSNAME..CRA.CARMALOG)
```

**Note:**

- Due to page width limitations, some lines in the sample wrapped onto the next line. All lines that start with an indentation should be added to the end of the previous line.
- To be able to create user-specific log files, this log file must be allocated using the `-COMMAND` statement.
- You can also allocate the log data sets in an allocation exec if you need more flexibility; for example, only send the log to a data set for specific users. For more information about allocation execs, see "(Optional) Custom allocation exec" on page 52.

If you are writing log files to SYSOUT, remember that SYSOUT allocated by z/OS UNIX processes is treated as special output in JES. This is similar to SYSOUT allocated by APPC transactions.

- While the CARMA server is still active, the output can be seen using the **DA** command in SDSF. The job will have the user's user ID followed by a random one-digit number as job name and an STC job ID. The user is the job owner.
- If the output was written to a HOLD output class, when the CARMA server ends, due to inactivity or the user ending the connection, the output can be seen using the **APPC ON** and **H ALL** commands in SDSF. The job name, job ID, and job owner remain the same. Each DD shows up as a separate spool file, without any indication which DD it is.
- JES Job Monitor can also show the output if `SEARCHALL=ON` is active in `FEJJCNFG` and the output resides on the spool in a HOLD output class. For more information about the `SEARCHALL` directive, see the *IBM Explorer for z/OS Host Configuration Guide (SC27-8437)*..

## CRASUB*, the batch submit server startup

RSE starts CLIST `CRASUB*`, which in turn submits an embedded JCL to create a valid environment to execute batch TSO and ISPF commands. Developer for z/OS uses this environment to run the CARMA server, `CRASERV`.

`CRASUB*` is located in `FEL.#CUST.CNTL`, unless you specified a different location when you customized and submitted the `FEL.SFELSAMP(FELSETUP)` job. For more details, see "Customization setup" on page 13.

**Note:** Changes are in effect for all CARMA servers that are started after the update.

Developer for z/OS provides multiple `CRASUB*` JCL jobs. Each of these sample files is pre-configured for a specific customization scenario:

- CRASUBCA is configured for batch startup with CA Endevor® SCM RAM.
- CRASUBMT is configured for batch startup with sample RAMs.

Figure 13 on page 43 shows a basic CRASUB* skeleton that includes ISPF services.

```
PROC 8 CRAPRM1 CRAPRM2 CRAPRM3 CRAPRM4 CRAPRM5 CRAPRM6 CRAPRM7 CRAPRM8
/* SET CRAPRM2=420
SUBMIT * END($$)
//CRA&CRAPRM1  JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//*
//RUN      EXEC PGM=IKJEFT01,DYNAMNBR=125,REGION=0M,TIME=NOLIMIT
//*
//* DD used by RAM
//*
//* DD used by CARMA server (CRASERV)
//STEPLIB  DD DISP=SHR,DSN=FEL.SFELLOAD
//CRADEF   DD DISP=SHR,DSN=FEL.#CUST.CRADEF
//CRAMSG   DD DISP=SHR,DSN=FEL.#CUST.CRAMSG
//CRASTRS  DD DISP=SHR,DSN=FEL.#CUST.CRASTRS
//CARMALOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*
//* DD used by ISPF (ISPSTART)
//ISPPROF  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(1,1,5))
//ISPCTL0  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(5,5))
//ISPCTL1  DD DISP=(NEW,DELETE,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(5,5))
//ISPMLIB  DD DISP=SHR,DSN=ISP.SISPMENU
//ISPPLIB  DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB  DD DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB  DD DISP=SHR,DSN=ISP.SISPTENU
//*
//* DD used by TSO (IKJEFT01)
//SYSPROC  DD DISP=SHR,DSN=#FEL.SFELPROC               * CRAALLOC
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
%CRAALLOC ISPSTART PGM(CRASERV) PARM(&CRAPRM1 &CRAPRM2 &STR(+)
&CRAPRM3 &STR(+)
&CRAPRM4 &STR(+)
&CRAPRM5 &STR(+)
&CRAPRM6 &STR(+)
&CRAPRM7 &STR(+)
&CRAPRM8 &STR(+)
)
$$
EXIT CODE(0)
```

*Figure 13. CRASUB*: CARMA startup using batch submit*

**Note:**

- If you alter the SYSTSIN data, ensure that there is at least one blank before the closing round bracket (“)”) of the PARM() statement to simplify processing of the string.
- You can add your own DD statements and data set concatenations to customize the CARMA TSO environment, thus mimicking a TSO logon procedure.
- You can use the CRAPRM* variables in the CLIST and JCL definitions.
  - &CRAPRM1.: Port number.
  - &CRAPRM8.: Site-specific value set by the user exit referenced by the `system.exit` directive in `CRASRV.properties`.
- Optionally, you can change CARMA's timeout value by uncommenting and modifying the SET CRAPRM2=420 line in the CRASUB* CLIST. The timeout value is the number of seconds that CARMA waits for the next command from the client. Setting a value of 0 results in the default timeout value, currently 420 seconds (7 minutes).
- Details of the CARMA startup process are shown in `rsecomm.log` when the server ends. For more information on setting the detail level of `rsecomm.log`, see the *IBM Explorer for z/OS Host Configuration Guide (SC27-8437)*..

## CARMA VSAM data sets

The CARMA server requires READ access to three VSAM data sets. The sample members to create and populate these VSAM data sets are located in FEL.#CUST.JCL, unless you specified a different

location when you customized and submitted the FEL.SFELSAMP(FELSETUP) job. For more details, see "Customization setup" on page 13.

**Note:**

- If you need to merge the definitions for a (custom) RAM into an existing VSAM configuration, see the FEL.#CUST.JCL(CRA#UADD) sample job. This job must be customized and submitted for each CARMA VSAM file that changes. For more information about the record structure used by the different CARMA VSAM files, see the *Common Access Repository Manager Developer's Guide* (SC23-7660).
- Use the FEL.#CUST.JCL(CRA#UQRY) sample job to extract the active definitions from a VSAM to a sequential data set.

### CRADEF, the configuration data set

This VSAM data set describes the functions that are supported by the defined RAMs. RAM developers require UPDATE access to this data set. The data set can be created by one of these sample jobs:

- CRA$VCAD populates the data set with CA Endevor® SCM RAM data.
- CRA$VDEF populates the data set with sample RAM data.

The mentioned sample jobs can be used to disable a defined RAM during VSAM creation. Doing so enables you to create a customized CARMA setup by using a single RAM definition file, which can be one provided by IBM or customized by your RAM developers.

### CRAMSG, the message data set

This VSAM data set holds messages issued by the CARMA server itself. The data set can be created by one of these sample jobs:

- CRA$VMSG populates the data set with generic server data.

### CRASTRS, the custom string data set

This VSAM data set holds the messages that are issued by the defined RAMs. RAM developers require UPDATE access to this data set. The data set can be created by one of these sample jobs:

- CRA$VCAS populates the data set with CA Endevor® SCM RAM data.
- CRA$VSTR populates the data set with sample RAM data.

## CARMA Repository Access Managers (RAMs)

A Repository Access Manager (RAM) is an Application Programming Interface (API) for a z/OS based Software Configuration Manager (SCM). In turn, Developer for z/OS or user-written applications can start a CARMA server, which loads the RAMs and provides a standard interface to access any supported SCM.

The CARMA server must be able to find the RAM load modules, either through LINKLIST or STEPLIB/TASKLIB.

The CRAR* RAM load modules that are provided by Developer for z/OS are located in FEL.SFELLOAD, and the sample source code and compile jobs are located in FEL.SFELSAMP, unless you used a different high level qualifier during the SMP/E install of Developer for z/OS.

The following sections have customization notes for the RAMs that are available with Developer for z/OS. The referenced sample members are located in FEL.#CUST.*, unless you specified a different location when you customized and submitted the FEL.SFELSAMP(FELSETUP) sample job. For more details, see "Customization setup" on page 13.

For in-depth knowledge of CARMA and for more information on the sample RAMs and sample source code provided, see *Common Access Repository Manager Developer's Guide* (SC23-7660).

## CA Endevor® SCM RAM

- The CA Endevor® SCM RAM is a production-type RAM.
- The CA Endevor® SCM RAM gives Developer for z/OS clients direct access to CA Endevor® SCM elements.
- The load module name is CRARNDVR.
- The CA Endevor® SCM RAM has many additional settings compared to a conventional CARMA setup. Use one of the preconfigured setups that support the CA Endevor® SCM RAM as starting point, and customize it to fit your needs.
- The CA Endevor® SCM RAM has multiple configuration files that can be customized. For more information, see "CRACFG, CRASCL, CRASHOW and CRATMAP, the CA Endevor® SCM RAM configuration files" on page 46.
- The CA Endevor® SCM RAM has an allocation exec, FEL.SFELPROC(CRANDVRA), that can be customized. See "CRANDVRA, the CA Endevor® SCM RAM initial allocation exec" on page 47 for more information.
- The CA Endevor® SCM RAM supports doing CA Endevor® SCM actions in batch mode, in the background. For more information, see "CA Endevor® SCM RAM batch actions" on page 48.

## CA Endevor® SCM packages RAM

- The CA Endevor® SCM packages RAM is a production-type RAM.
- The CA Endevor® SCM packages RAM gives Developer for z/OS clients direct access to CA Endevor® SCM packages.
- The load module name is CRARPKGS.
- The CA Endevor® SCM packages RAM does not have customizable settings, and must be used in combination with the CA Endevor® SCM RAM.

## PDS RAM

- The PDS RAM is a sample RAM. Do not use in a production environment.
- The PDS RAM gives a data set list similar to **MVS Files -> My Data Sets** in the Remote Systems view.
- The load module name is CRARPDS.
- The PDS RAM requires that ISPF services be available.
- The PDS RAM requires an additional VSAM data set to be allocated to DD CRARAM1. This VSAM data set can be allocated and primed with the FEL.#CUST.JCL(CRA#VPDS) sample job. For customization instructions, see the documentation within the member.
- Source code and compile jobs are available in FEL.SFELSAMP. For more information, see *Common Access Repository Manager Developer's Guide* (SC23-7660).

## Skeleton RAM

- The skeleton RAM is a sample RAM. Do not use in a production environment.
- The skeleton RAM gives a functional framework that can be used as starting point to develop your own RAM.
- The load module name is CRARTEST.
- Source code and compile jobs are available in FEL.SFELSAMP. For more information, see *Common Access Repository Manager Developer's Guide* (SC23-7660).

## SCLM RAM

- The SCLM RAM is a sample RAM. Do not use in a production environment.
- The SCLM RAM gives a basic entry into SCLM, ISPF's Software Configuration Manager. This RAM is not enabled by default.

- The load module name is CRARSCLM.
- The SCLM RAM needs the ISPF services to be available.
- The SCLM RAM requires an additional VSAM data set to be allocated to DD CRARAM2. This VSAM data set can be allocated and primed with the FEL.#CUST.JCL(CRA#VSLM) sample job. For customization instructions, see the documentation within the member.
- The SCLM RAM requires the various user-specific data sets to exist. Customize FEL.#CUST.JCL(CRA#ASLM) to allocate these data sets. For customization instructions, see the documentation within the member. Each user must submit CRA#ASLM once before using CARMA with the SCLM RAM. Failing to do so will result in an allocation error.
- The SCLM RAM is not enabled by default. To enable the RAM, it must be defined in the CARMA VSAM data sets referenced by DD CRADEF and CRASTRS. Use the FEL.#CUST.JCL(CRA#UADD) sample job to merge FEL.SFELVSM2(CRA0SLMD) into CRADEF and FEL.SFELVSM2(CRA0SLMS) into CRASTRS. For customization instructions, see the documentation within the member.
- Source code and compile jobs are available in FEL.SFELSAMP. For more information, see *Common Access Repository Manager Developer's Guide* (SC23-7660).

# CRACFG, CRASCL, CRASHOW and CRATMAP, the CA Endevor® SCM RAM configuration files

The following CA Endevor® SCM RAM-specific CARMA components can be customized, regardless of the chosen server startup method. The sample members referenced below are located in FEL.#CUST.PARMLIB, unless you specified a different location when you customized and submitted the FEL.SFELSAMP(FELSETUP) job. For more details, see "Customization setup" on page 13.

## CRACFG, CA Endevor® SCM RAM interaction with the SCM

CRACFG specifies how the CA Endevor® SCM RAM interacts with CA Endevor® SCM. Refer to the documentation within the member for customization instructions if you want to change the defaults.

```
# ENTRY-STAGE-COPY-MODE = RETRIEVE-ADD
# ALTERNATIVE-ALLOC = YES
# PACKAGE-EDITING-OPTION = READONLY
# PACKAGE-EDITING-OPTION = DISABLED
# SCL-REQUIRED = YES
# SCL-DATASET-TEMPLATE = FEL.#CUST.PARMLIB(CRASCL)
# DYNAMIC-VB-DATASET-ALLOC = YES
# DYNAMIC-FB-DATASET-ALLOC = YES
# DATASET-ALLOC-OVERRIDE = SPACE(5,30) TRACK UNIT(SYSALLDA)
# BYPASS-RESOURCE-LOCKING = YES
```

*Figure 14. CRACFG - CA Endevor® SCM RAM interaction with the SCM*

## CRASCL, CA Endevor® SCM RAM template SCL

CRASCL is a template SCL (Software Control Language) that can limit which actions and options are allowed when CA Endevor® SCM Packages are processed.

When used, only actions and options explicitly listed in the template SCL are allowed to be specified in the Developer for z/OS Packages Editor. Refer to the documentation within the member for customization instructions if you want to change the defaults.

## CRASHOW, CA Endevor® SCM RAM default filters

CRASHOW defines default filters for CA Endevor® SCM environments, systems, and so forth. Refer to the documentation within the member for customization instructions if you want to change the defaults.

```
ENV=*
TOENV=
STGID=*
TOSTGID=
SYS=*
SUBSYS=*
ELEM=*
TOELEM=
TYPE=*
#FILTER-DEP=YES
```

*Figure 15. CRASHOW - CA Endevor® SCM RAM default filters*

**Note:** FILTER-DEP is not a common CA Endevor® SCM variable, but a Developer for z/OS specific variable that controls dependency scans for elements with footprint references to other CA Endevor® SCM repository locations.

## CRATMAP, the CA Endevor® SCM RAM file extension mappings

CRATMAP overrides the CA Endevor® SCM type to file extension mappings. If you want to change the defaults, see the customization instructions in the documentation within the member.

```
# *       = cbl
# COBOL   = cbl
# COPY    = cpy
# ASM     = asm
# MACRO   = asm
# PROCESS = jcl
```

*Figure 16. CRATMAP: CA Endevor® SCM RAM default filters*

# CRANDVRA and CRADYNDA, the CA Endevor® SCM RAM allocation execs

The following CA Endevor® SCM RAM-specific CARMA components can be customized, regardless of the chosen server startup method.

You can customize a copy of these allocation REXX execs if certain defaults, such as the data set name, do not match your site standards. The execs are located in FEL.SFELPROC, unless you used a different high-level qualifier during the SMP/E install of Developer for z/OS.

For customization instructions, see the documentation within the member. For more information about allocation execs, see "(Optional) Custom allocation exec" on page 52.

**Note:** You should copy the sample allocation REXX to a new data set and customize this copy to avoid overwriting it when applying maintenance. When you do this, you must update the reference to SFELPROC in the SYSEXEC DD of your chosen CARMA startup method to match your new data set name.

## CRANDVRA, the CA Endevor® SCM RAM initial allocation exec

Both the batch submit and the CRASTART startup method call the CRANDVRA REXX exec to allocate user-specific data sets used by CA Endevor® SCM RAM. The allocations are done in a separate exec, because an exec allows more flexibility than what is possible within the batch submit CRASUBCA JCL and the CRASTART `crastart.endevor.conf` configuration file. The allocation exec is also responsible for calling the optional user exit.

| DD | Data set name | Type |
|---|---|---|
| DEPEND | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.DEPEND | Permanent |
| BROWSE | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.BROWSE | Temporary |
| BROWSEV | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.BROWSEV | Temporary |
| ENHCEDIT | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.ENHCEDIT | Temporary |
| ENHCEDITV | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.ENHCEDITV | Temporary |

| DD | Data set name | Type |
|---|---|---|
| C1PRINT | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.LISTING | Temporary |
| SPCLLIST | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.SPCLLIST | Temporary |
| PKGSCLS | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.PKGSCLS | Temporary |
| CRABJCLO | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.CRABJCLO | Temporary |
| CRAPARM | &SYSPREF..&SYSUID..&SYSNAME..CRA$NDVR.CRAPARM | Temporary |

### CRADYNDA, the CA Endevor® SCM RAM reallocation exec

CA Endevor® SCM can work with variable blocked data sets with various record lengths, and requires that Developer for z/OS uses identical allocations for certain actions. Both the batch submit and the CRASTART startup method call the CRADYNDA REXX exec to allocate a work data set with the correct attributes.

## CA Endevor® SCM RAM batch actions

Normally, CA Endevor® SCM actions such as “Generate Element” are executed “online”, in the CARMA server address space. This behavior causes problems if your CA Endevor® SCM procedures call TSO, because TSO is already active and that means that the required DDs such as SYSTSIN and SYSTSPRT are in use.

To resolve this problem, the CA Endevor® SCM RAM supports “batch actions”. When batch-actions is enabled, the CA Endevor® SCM RAM submits a customizable batch job to perform actions like “Generate Element”. Using a batch job results in the allocation of DDs such as SYSTSIN and SYSTSPRT by your CA Endevor® SCM procedures, because the submitted JCL does not require TSO to be active.

CA Endevor® SCM RAM batch-actions are the Developer for z/OS equivalent of background CA Endevor® SCM actions.

When a request is issued to execute an action that is supported by batch-actions, the CA Endevor® SCM RAM checks for the existence of the CRABCFG DD, in CRASUBCA or `crastart.endevor.conf`, and checks that the setup behind this DD is valid. If CRABCFG exists and the setup is valid, the action is performed in batch. If CRABCFG does not exist, the action is performed online. Developer for z/OS clients have the facility to override this behavior.

For example:

```
//* uncomment CRABCFG and CRABSKEL to use batch actions
//*CRABCFG  DD DISP=SHR,DSN=FEL.#CUST.PARMLIB(CRABCFG)
//*CRABSKEL DD DISP=SHR,DSN=FEL.#CUST.CNTL
```

**Note:**

- The TSO-free environment is available only for selected CA Endevor® SCM actions. Batch-actions does not support a TSO-free environment outside this scope.
- The CRABCFG configuration file documents which CA Endevor® SCM actions are supported.
- A functional sample job, `FEL.#CUST.CNTL(CRABATCA)`, is provided to execute the batch actions, but the intent of batch-actions is that this sample is customized to start your current CA Endevor® SCM procedures.
- Ensure that there are sufficient JES initiators available in the class used to submit the batch-action JCLs.
- When using JES in a SYSPLEX environment, ensure that the job runs on the current system, or that the completion information is routed back to the system hosting Developer for z/OS, so that the CA Endevor® SCM RAM can check the status.
- The Developer for z/OS client can provide a customized JOB card and additional JCL statements to the batch-action JCL before submission.

- The batch-action JCL has access to the CRAPRM* variables through generated SET statements. This includes CRAPRM8, the variable reserved for site-specific data, which can be altered by the user exit referenced by the `system.exit` directive in `CRASRV.properties`.

## CRABCFG, the CA Endevor® SCM RAM batch-action configuration

CRABCFG defines the configuration variables related to CA Endevor® SCM RAM batch-actions.

CRABCFG is located in `FEL.#CUST.PARMLIB`, unless you specified a different location when you customized and submitted the `FEL.SFELSAMP(FELSETUP)` job. For more details, see "Customization setup" on page 13.

See the following CRABCFG sample file, which must be customized to match your system environment. Comment lines start with a number sign (#) when using a US code page. Comments behind a directive and its assigned value are supported. Spaces around the equal sign (=) are supported. Line continuations are not supported.

**Note:** Changes are in effect for all CARMA servers that are started after the update.

```
# Location of batch action JCL
SKELETON-DD = CRABSKEL
#
# batch action JCL members within SKELETON-DD
DEFAULT-JOBCARD  = CRABJOBC
ADD-ELEMENT      = CRABATCA
DELETE-ELEMENT   = CRABATCA
GENERATE-ELEMENT = CRABATCA
MOVE-ELEMENT     = CRABATCA
RETRIEVE-ELEMENT = CRABATCA
PRINT-ELEMENT    = CRABATCA
PRINT-MEMBER     = CRABATCA
SIGNIN-ELEMENT   = CRABATCA
TRANSFER-ELEMENT = CRABATCA
#
# Command substitution key within batch action JCL
BSTIPT01-KEY = <CRA_BSTIPT01>
```

*Figure 17. CRABCFG: CA Endevor® SCM RAM batch-action configuration*

**SKELETON-DD**
Name of the DD statement that references one or more PDS(E) data sets that hold the batch-action skeleton JCLs. The sample value is CRABSKEL. Can be changed if needed. This DD must be defined to the CARMA server in CRASUBCA or `crastart.endevor.conf`.

**DEFAULT-JOBCARD**
Name of the member holding a default JOB card. If not overruled by a user-specific JOB card stored on the Developer for z/OS client, this default JOB card is used to substitute the <JOBCARD> key in a skeleton JCL. Can be changed if needed.

**GENERATE-ELEMENT and other CA Endevor® SCM actions**
The key names represent the CA Endevor® SCM actions that are supported by batch-action and cannot be changed. The value assigned to each key is the member name of the related skeleton JCL. The sample value is CRABATCA for all keys. Can be changed if needed.

**BSTIPT01-KEY**
Substitution key for the actual CA Endevor® SCM command string. The sample value is <CRA_BSTIPT01>. Can be changed if needed. The first occurrence, but not in a comment, of this substitution key within the skeleton JCL is replaced by the command string that instructs CA Endevor® SCM to do the requested action against the requested element.

## CRABATCA, the CA Endevor® SCM RAM batch action JCL

CRABATCA is a sample skeleton JCL used for batch-actions. To change the defaults, see the customization instructions in the documentation within the member.

CRABATCA is located in FEL.#CUST.CNTL, unless you specified a different location when you customized and submitted the FEL.SFELSAMP(FELSETUP) job. For more details, see “Customization setup” on page 13.

Changes are active for all new invocations. No server restart is needed.

```
//<JOBCARD>
/*JOBPARM SYSAFF=*
//*
//<SET_CRAPRM>
//*
//CRABATCA EXEC PGM=NDVRC1,DYNAMNBR=1500,REGION=4096K,PARM='C1BM3000'
//STEPLIB  DD DISP=SHR,DSN=CA.NDVR.CSIQAUTU
//         DD DISP=SHR,DSN=CA.NDVR.CSIQAUTH
//CONLIB   DD DISP=SHR,DSN=CA.NDVR.CSIQLOAD
//C1MSGS1  DD SYSOUT=*
//C1MSGS2  DD SYSOUT=*
//C1PRINT  DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133)
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYMDUMP  DD DUMMY
//SYSIN    DD DUMMY
//BSTIPT01 DD *
SET STOPRC 16 .
<CRA_BSTIPT01>
//*
```

*Figure 18. CRABATCA: CA Endevor® SCM RAM batch-action JCL*

## CRABJOBC, the CA Endevor® SCM RAM batch action JOB card

CRABJOBC is a sample default JOB card used for batch-action skeleton JCL that specifies the <JOBCARD> key. To change the defaults, see customization instructions in the documentation within the member.

CRABJOBC is located in FEL.#CUST.CNTL, unless you specified a different location when you customized and submitted the FEL.SFELSAMP(FELSETUP) job. For more details, see “Customization setup” on page 13.

Changes are active for all new invocations. No server restart is needed.

```
//<USERID>B JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//*PROCS JCLLIB ORDER=(COBOL.V5R1M0.SIGYPROC,CBC.SCCNPRC)
```

*Figure 19. CRABJOBC: CA Endevor® SCM RAM batch-action JOB card*

# CRAALLOC, the custom RAM allocation exec

Both the batch submit and the CRASTART startup method call the CRAALLOC REXX exec to allocate user-specific data sets that can be used by a user-written RAM. The allocations are done in a separate exec, because an exec allows more flexibility than what is possible within the batch submit CRASUBMT JCL and the CRASTART `crastart.conf` configuration file. The allocation exec is also responsible for calling the optional user exit.

| DD | Data set name | Type |
|---|---|---|
| CRAPARM | &SYSPREF..&SYSUID..&SYSNAME..CRA$CUST.CRAPARM | Temporary |

You can customize a copy of this allocation REXX exec if certain defaults, such as the data set name, do not match your site standards. CRAALLOC is located in FEL.SFELPROC, unless you used a different high-level qualifier during the SMP/E install of Developer for z/OS.

For customization instructions, see the documentation within the member. For more information about allocation execs, see (Optional) Custom allocation exec.

**Note:** You should copy the sample allocation REXX to a new data set and customize this copy to avoid overwriting it when applying maintenance. When you do this, you must update the reference to SFELPROC in the SYSEXEC DD of your chosen CARMA startup method to match your new data set name.

## CARMA return codes

CARMA can report various error codes to the client or in the host system logs. The details that are provided with the error, and the information in Table 12 on page 51, can help you locate the error and work towards a resolution.

*Table 12. CARMA return codes*

| Error range | Error type |
|---|---|
| 4-99 | Generic CARMA errors |
| 100-199 | Generic RAM errors |
| 200-399 | CRASERV (CARMA server) errors |
| 400-499 | RSE (CARMA miner) errors |
| 500-899 | RAM-specific errors |
| 900-999 | TSO and TCP/IP errors |

Some common return codes are these:

- 220: CARMA server ends due to inactivity timeout. This is not an error.
- 990: CARMA server is unable to connect to the port on which CARMA miner is listening.

# (Optional) Supporting multiple RAMs

CARMA has the facility for defining multiple RAMs and running them concurrently. However, because there is only one CARMA server active for a user, even when there are multiple RAMs, some configuration changes might be required to make this setup work.

RAMs are defined by a RAM developer in the CARMA configuration VSAM data set, CRADEF. During startup, the CARMA server, CRASERV, identifies all of the defined RAMs and sends the information to the CARMA client. The user can then select one or more RAMs, which is loaded into the CARMA server.

Because RAMs are active as plug-ins of the CARMA server, ensure that all prerequisites, such as data set allocations, for each of the RAMs are available in the address space of the CARMA server. This requirement might need changes to the CARMA configuration samples, such as CRASUBMT or `crastart.conf`, which are included with Developer for z/OS.

## Example

In the following example, you start from an existing setup with the CA Endevor® SCM RAM, using the CRASTART startup method, and add the sample PDS RAM.

Definitions for the CA Endevor® SCM RAM:

- `FEL.SFELVSM2(CRA0VCAD)`: CRADEF definitions
- `FEL.SFELVSM2(CRA0VCAS)`: CRASTRS definitions
- `/etc/zexpl/crastart.endevor.conf`: CRASTART configuration file

Definitions for the PDS RAM:

- `FEL.SFELVSM2(CRA0VDEF)`: CRADEF definitions
- `FEL.SFELVSM2(CRA0VSTR)`: CRASTRS definitions
- `FEL.#CUST.CRARAM1`: CRARAM1 definitions

The process starts with a RAM developer gathering the data and information needed by the system programmer to complete the setup.

1. Extract the data that is specific for the PDS RAM from the SFELVSM2 members. These members hold definitions for all sample RAMs, not just the PDS RAM.
2. Merge this data with the CA Endevor® SCM RAM SFELVSM2 members.
3. Create a list of PDS RAM-specific prerequisites:
   - DD CRARAM1, linked to FEL.#CUST.CRARAM1
   - TSO environment

The system programmer then uses this data to create the updated CARMA VSAM data sets and uses the prerequisite information to create a CRASTART configuration file that is capable of supporting both RAMs.

1. Use the combined data as input for the CRA$VDEF and CRA$VSTR jobs to create the updated CARMA configuration and custom information VSAM data sets, CRADEF and CRASTRS. The CRAMSG VSAM is specific for the CARMA server, and thus identical for both RAMs.
2. Add a CRARAM1 definition to `crastart.endevor.conf`:

   ```
   CRARAM1 = FEL.#CUST.CRARAM1
   ```

3. Verify the PROGRAM statement in `crastart.endevor.conf` to ensure that it is capable of providing the environment needed by both RAMs.
4. 
   ```
   PROGRAM=IKJEFT01 %CRANDVRA NDVRC1 PGM(CRASERV)
     PARM(&CRAPRM1. &CRAPRM2.)
   ```

   - IKJEFT01: TSO, used to allow certain authorized calls in a non-authorized environment, and used as environment to run the CA Endevor® SCM RAM pre-allocation exec.
   - %CRANDVRA: CA Endevor® SCM RAM pre-allocation exec, located in FEL.SFELPROC, that allocates temporary user-specific working data sets.
   - NDVRC1: CA Endevor® back end, which has a built-in mechanism to execute TSO and ISPF commands.
   - PGM(CRASERV): Command to start a CARMA server, in ISPF command format.
   - PARM(&CRAPRM1. &CRAPRM2.): Parameters for CRASERV, in ISPF command format. &CRAPRM1 is the port to be used and &CRAPRM2 is the timeout value.

The CA Endevor® SCM RAM is active in an ISPF environment, which implies that the TSO environment required by the PDS RAM is also available.

## (Optional) Custom allocation exec

All CARMA server startup methods have limitations regarding data set allocation. For example, TSO prefix substitution is not available in JCL or CRASTART.

However, by creating an exec that is called after TSO or ISPF starts, and before CARMA is started, you can use the whole range of variables and services available in TSO or ISPF to do the required allocations.

Developer for z/OS uses an allocation exec in each of the pre-configured setups described earlier in this chapter. FEL.SFELPROC(CRANDVRA), the allocation exec for CA Endevor® SCM RAM and FEL.SFELPROC(CRAALLOC), the allocation exec for custom RAMs, The exec allocates cataloged temporary data sets that have the user's TSO prefix as high-level qualifier. The allocation exec is also responsible for calling the optional user exit.

Customization instructions are documented within the exec. Changing the allocation exec is supported, but not advised, as customizations must be redone when PTF service updates the exec. If possible, use the CARMA user exit instead, which is described in "(Optional) CARMA user exit" on page 53.

**Note:**

- When updating an allocation exec, ensure you do not destroy allocations done earlier in the CARMA startup process by CRASTART or your startup JCL.
- Output generated by the allocation exec is shown in DD SYSTSPRT of the CARMA server.

When updating an allocation exec, ensure you do not destroy allocations done earlier in the CARMA startup process by CRASTART or your startup JCL.

The following samples show how to start an allocation exec that requires only TSO.

crastart*.conf

```
SYSPROC = my.exec.library
PROGRAM = IKJEFT01 %myexec ISPSTART PGM(CRASERV) PARM(&CRAPRM1. &CRAPRM2. )
```

CRASUB*

```
//SYSPROC  DD DISP=SHR,DSN=my.exec.library
//SYSTSIN  DD *
%myexec ISPSTART PGM(CRASERV) PARM(&CRAPRM1. &CRAPRM2. )
//*
```

# (Optional) CARMA user exit

CARMA supports the invocation of a user exit to allow for specialized initialization during startup and specialized cleanup during shutdown of the CARMA server. The usage of a user exit allows you to provide site-specific values to CARMA, and reduces the need to alter the allocation exec, which is maintained by PTF service.

CARMA allows you to specify two user exits. Each of these user exits are invoked twice: once just before a major resource allocation in CARMA server startup, like address space creation, and once right after the allocated resources are released, like address space termination.

User exits are activated with the `*.exit` directives in `CRASRV.properties`. `CRASRV.properties` is located in `/etc/zexpl/`, unless you specified a different location when you customized and submitted `FEL.SFELSAMP(FELSETUP)` job. For more details, see Customization setup.

**Note:** CARMA executes the provided user exit code without question. To maintain a secure environment, it is important that you limit update access to the user exit code.

## system.exit – exit points for address space creation and termination

This z/OS UNIX-based user exit is invoked just before the address space is created in which TSO, and eventually the CARMA server, is started. The exit is able to alter most variables that influence the startup of the CARMA server. This includes the ability to provide a site-specific value in variable CRAPRM8, which can be used during address space creation, CARMA server startup, and throughout the active life of the CARMA server. The same user exit is invoked again after the address space is terminated. If the first invocation ends with return code 99 or higher, CARMA server startup is halted at that point.

This exit can alter CRAPRM8 before it is used in `crastart*.conf` and CRASUB* during address space creation, which allows you to define a customized STEPLIB definition. For example, based on the connection name provided by the client, you can select a different release of your SCM.

A sample user exit is provided as `/usr/lpp/IBM/zee/samples/craexit.rex`. Unless you used a different path during the SMP/E install of Developer for z/OS.

- craexit.rex: A sample exit that alters CRAPRM8 based on the client connection name to allow for different STEPLIB definitions, and thus different CA Endevor® SCM instances to be selected without the usage of CA's ENUXSITE program.

The sample user exit documents in detail the startup arguments passed to the user exit:

| Startup argument | Description |
|---|---|
| (STARTUP) \| (ENDING) | Indicator whether the exit invocation is before or after CARMA server invocation. |

| Startup argument | Description |
|---|---|
| EXIT_RC=rc | Return code of the previous invocation of the exit.<br>rc Is always 0 during (STARTUP) invocation. |
| CARMA_RC=rc | Return code of the invocation of CARMA server.<br>rc Is always 0 during (STARTUP) invocation. |
| (CLIST) \| (CRASTART) | CARMA server startup method indicator. |
| ... | Variables used for CARMA server address space creation. These are dependent on the startup method.<br>• CLIST: For example,<br>`STARTUP=FEL.#CUST.CNTL(CRASUBCA)`<br>CRASTART: For example,<br>`STARTUP=/usr/lpp/IBM/zee/bin/CRASTART`<br>`FILE=/etc/zexpl/crastart.endevor.conf`<br>`SYSLOG=P TASKLIB=TASKLIB BACKGROUND=0` |
| -!- | Divider between address space and startup variables. |
| ... | CARMA server startup variables. These variables are known as CRAPRM1 to CRAPRM8. For example, `1312 420 NOEXIT CLIENT=14.0.3.idz140030-I20161120_1440 CONNECTION=Endevor18 . . .` |

The user exit can, besides a return code, return a string holding new values for most of the variables it received as startup argument:

| Startup argument | Description |
|---|---|
| CRAEXIT_RETURN | Keyword marking the string as return data. |
| ... | Variables used for CARMA server address space creation. These are dependent on the startup method.<br>• CLIST: For example,<br>`STARTUP=FEL.#CUST.CNTL(CRASUBCA)`<br>• CRASTART: For example:<br>`STARTUP=/usr/lpp/IBM/zee/bin/CRASTART`<br>`FILE=/etc/zexpl/crastart.endevor.conf`<br>`SYSLOG=P TASKLIB=TASKLIB BACKGROUND=0` |
| -!- | Divider between address space and startup variables. |

| Startup argument | Description |
|---|---|
| ... | CARMA server startup variables. These variables are known as CRAPRM1 to CRAPRM8. For example, `1312 420 NOEXIT CLIENT=14.0.3.idz140030-I20161120_1440 CONNECTION=Endevor18 . . CAI.NDVR18.REGIONH` |

Output generated by the user exit is visible in `rsecomm.log`, when debug level 2 is active for `rsecomm.log`.

### user.exit – exit points for server startup and shutdown

This MVS based user exit is invoked by the allocation exec, and is executed twice. The initialization invocation is after the allocation of the temporary data sets and before the CARMA server is invoked. The cleanup invocation is after the CARMA server ended and before the temporary files are removed. If the first invocation ends with return code 99 or higher, CARMA startup is interrupted. This implies that neither CARMA server nor the second invocation of this user exit is executed.

Sample user exits are provided as FEL.SFELSAMP(CRAEXIT*), unless you used a different high-level qualifier during the SMP/E install of Developer for z/OS.

- CRAEXIT: A sample exit that allocates different CARMA VSAM data sets based on the client version.
- CRAEXIT2: A sample exit that invokes different CA Endevor® SCMs based on the client connection name.

The sample user exits document in detail the startup arguments passed to the user exit:

| Startup argument | Description |
|---|---|
| (STARTUP) \| (ENDING) | Indicator whether the exit invocation is before or after CARMA server invocation. |
| EXIT_RC=rc | Return code of the previous invocation of the exit.<br>rc Is always 0 during (STARTUP) invocation. |
| CARMA_RC=rc | Return code of the invocation of CARMA server.<br>`rc` is always 0 during (STARTUP) invocation. |
| ... | CARMA server startup command and startup arguments. For example, `ISPSTART PGM(CRASERV) PARM(1312 420 EXIT=FEL.#CUST(CRAEXIT) CLIENT=14.0.3.idz140030-I20161120_1440 CONNECTION=Endevor18 . . . )` |

Output generated by the user exit is shown in DD SYSTSPRT of the CARMA server.

## (Optional) IRXJCL versus CRAXJCL

If the CARMA server is started using TSO (IKJEFTxx), problems might occur if your RAMs call services which in turn call the IRXJCL REXX batch interface. The problem can occur when the processors called by the RAM previously ran either without TSO, or only in online TSO, and dynamically allocates DD SYSTSIN or SYSTSPRT. A sample program, CRAXJCL, is provided to work around this problem.

Your processor might fail if it attempts to allocate SYSTSIN or SYSTSPRT, which is required for IRXJCL, because batch TSO required for CARMA already has those DD names allocated and open. The CRAXJCL replacement module attempts to allocate SYSTSIN and SYSTSPRT to DUMMY but ignores the errors which occur if the allocations fail. It then calls IRXJCL to do the actual work.

This means that when your processors run in a CARMA environment started by TSO, the allocations to SYSTSIN and SYSTSPRT are the same as those used by CARMA. When the processors are run outside of TSO/CARMA, the SYSTSIN and SYSTSPRINT allocations are created by CRAXJCL. Therefore, your processors must not rely on the contents of the data set allocated to SYSTSIN.

It is assumed that calls to IRXJCL use the PARM field to pass the REXX name and startup parameters, as documented in *TSO/E REXX Reference* (SA22-7790). This means that SYSTSIN can safely be used by CARMA. Any output sent to SYSTSPRT by IRXJCL is written in CARMA's log.

Processors that call the CRAXJCL replacement module should not attempt to allocate DD SYSTSIN or SYSTSPRT before calling CRAXJCL.

## Create CRAXJCL

The CRAXJCL replacement module is provided in source format because you must customize it to specify the specific allocations to use for SYSTSPRT. The allocation for SYSTSIN should usually be to a dummy data set.

Sample assembler source code and a sample compile/bind job are available as `FEL.#CUST.ASM(CRAXJCL)` and `FEL.#CUST.JCL(CRA#CIRX)`, unless you specified a different location when you customized and submitted `FEL.SFELSAMP(FELSETUP)` job. For more details, see "Customization setup" on page 13.

Customize the `CRAXJCL` assembler source code as needed, using the documentation within the member. Afterward, customize and submit the `CRA#CIRX` JCL to create the CRAXJCL load module. For customization instructions, see the documentation within the member.

If needed, you can rename IRXJCL to something else. Adjust the CRAXJCL source to call this new name for IRXJCL and compile it, and then rename the CRAXJCL load module to IRXJCL. This setup might be easier than changing all your calls to IRXJCL.

# Chapter 4. Host-based code analysis

Similar to the Developer for z/OS client, the Developer for z/OS host supports running code analysis tools, which are provided as a separate product, IBM z/OS Source Code Analysis, FMID HAKGxxx. A benefit of doing code analysis on the host is that it can be integrated in your daily batch processing.

The following code analysis tools are available on the host:

- Code review: Using rules with different severity levels, code review scans source code and reports rule violations.
- Code coverage: Analyze a running program and generate a report of lines that are executed, compared to the total number of executable lines.

## Requirements and checklist

You do not need assistance of other administrators to start using host-based code analysis tools at your site, but you must perform the following tasks. Unless otherwise indicated, all tasks are mandatory.

1. Install z/OS Source Code Analysis, as documented in *Program Directory for IBM z/OS Source Code Analysis* (GI13-2864). When using the provided defaults, the product is installed using high-level qualifier AKG and z/OS UNIX path `/usr/lpp/IBM/akg`.
2. Create customizable copies of the provided samples by customizing and submitting `AKG.SAKGSAMP(AKGSETUP)`. This job performs the following tasks:
   - Create `AKG.#CUST.PROCLIB` and populate it with sample `SYS1.PROCLIB` members.
   - Create `AKG.#CUST.JCL` and populate it with sample configuration JCL.

**Note:** IBM z/OS Debugger is a prerequisite for the Code coverage component. The IBM z/OS Debugger FMID (HADRxxx) is provided together with the IBM z/OS Source Code Analysis FMID.

## Code review

Code review scans source code and reports rule violations, using rules with different severity levels. The tool comes with rule providers for COBOL and PL/I, but other rule providers can be added.

z/OS Source Code Analysis provides a sample procedure, AKGCR, to simplify the calling of code review services in batch mode (also called host-based code review). AKGCR is found in `AKG.#CUST.PROCLIB`, unless you specified a different location when you customized and submitted the `AKG.SAKGSAMP(AKGSETUP)` job.

Customize the sample procedure, `AKG.#CUST.PROCLIB(AKGCR)`, as described within the member, and copy it to `SYS1.PROCLIB`.

If the AKGCR procedure cannot be copied into a system procedure library, ask the Developer for z/OS users to add a JCLLIB card right after the JOB card to their calling job.

```
//MYJOB    JOB <job parameters>
//PROCS    JCLLIB ORDER=(AKG.#CUST.PROCLIB)
```

**Note:** The AKGCR procedure requires data files to be created on the local workstation and copied in binary format to a directory in z/OS UNIX System Services. For more information about creating data files for the AKGCR procedure, see Creating and uploading data files.

### Modify code review processing

Developer for z/OS Code Review allows for third-party code to be part of the review process. For example, you can provide a rule provider to analyze C/C++ code, or you can enhance the Cobol rule provider to recognize site-specific coding conventions.

© Copyright IBM Corp. 2000, 2024

Host-based code review is an Eclipse process, just like the Developer for z/OS client. Therefore, the enhancements done by your development support team for code review on the client can be reused on the host.

The enhancements will consist of Eclipse plugins or Eclipse features. In order to activate them, you must make them available to the existing code, as documented in the AKGCRADD configuration job. AKGCRADD is in AKG.#CUST.JCL, unless you specified a different location when you customized and submitted the AKG.SAKGSAMP(AKGSETUP) job.

# Code coverage

Code coverage analyzes a running program and generates a report of lines that are executed, compared to the total number of executable lines. Note that code coverage sets up a TCP/IP connection, using an ephemeral port, with IBM z/OS Debugger.

z/OS Source Code Analysis and z/OS Debugger provide two ways to invoke Code coverage in batch mode, A sample JCL procedure, to process a single program run, and a set of scripts to start and stop a permanently active code coverage collector that can process multiple program runs.

## Single Code coverage invocation

The AKGCC sample procedure provides a method to start a Code coverage collector, have it analyze a single program run, stop the collector and archive the results for later usage.

AKGCC is in AKG.#CUST.PROCLIB, unless you specified a different location when you customized and submitted the AKG.SAKGSAMP(AKGSETUP) job.

Customize the sample procedure, AKG.#CUST.PROCLIB(AKGCC), as described within the member, and copy it to SYS1.PROCLIB.

If the AKGCC procedure cannot be copied into a system procedure library, ask the Developer for z/OS users to add a JCLLIB card right after the JOB card to their calling job.

```
//MYJOB    JOB <job parameters>
//PROCS    JCLLIB ORDER=(AKG.#CUST.PROCLIB)
```

## Multiple Code coverage invocations

Depending on how your software development process is set up, the convenience of having the AKGCC procedure take care of everything might not outweigh the resource and time usage to start a Code coverage collector for each program analysis.

**Note:** In version 15.0, this feature is moved to the z/OS Debugger code.

z/OS Debugger provides the `ccstart` script to start a Code coverage collector which remains active. This collector can then be used in multiple Code coverage invocations. The `ccstop` script can be used to stop the collector.

These scripts (`ccstart`, `ccstop`, and `codecov`) are located in `/usr/lpp/IBM/debug/headless-code-coverage/bin/` if you installed z/OS Debugger in the default location. For usage of these scripts, see Running code coverage in headless mode by using a daemon in the IBM Developer for z/OS Knowledge Center.

The following is a generic usage scenario:

1. Invoke `ccstart` with option to use a fixed port number and option to redirect the output to a known location.
2. Invoke, as often as needed, programs to be analyzed with startup option.TEST(,,,TCPIP&<hostip>:<port>).
3. Invoke `ccstop`.
4. Direct the Developer for z/OS client to the output location to see the reports.

## Code coverage output

The output of code coverage is intended to be imported into a Developer for z/OS client, and is therefore written to a z/OS UNIX file. Code coverage is also able to use the results of a previous run and combine them with the results of the current run, resulting in a single report that covers multiple code paths.

For these reasons, z/OS Source Code Analysis does not attempt to remove the output of a code coverage run, and the output will thus accumulate over time.

z/OS UNIX provides a shell script, `skulker`, that deletes files based on the directory they are in and their age. Combined with the z/OS UNIX `cron` daemon, which runs commands at specified dates and times, you can set up an automated tool that periodically cleans out targeted directories. Refer to *UNIX System Services Command Reference* (SA22-7802) for more information about the `skulker` script and the `cron` daemon.

# Chapter 5. Other customization tasks

This section combines various optional customization tasks. To configure the required service, follow the instructions in the appropriate section.

Customizations to Developer for z/OS configuration files:

- “include.conf, Forced includes for C/C++ content assist” on page 61

Developer for z/OS related customizations to or for other products:

- “z/OS UNIX subprojects” on page 62
- “Include preprocessor support” on page 63
- “ISPF EDIT macro support” on page 63
- “ISPF PACK support” on page 63
- “xUnit support for Enterprise COBOL and PL/I” on page 64
- “xUnit support for CICS applications (ZUnit)” on page 65
- “Enterprise Service Tools support” on page 69
- “CICS bidirectional language support” on page 69
- “Diagnostic IRZ messages for Enterprise Service Tools” on page 70

## `include.conf`, Forced includes for C/C++ content assist

This customization task does not require assistance, special resources, or special customization tasks.

Content assist for C/C++ can use the definitions in `include.conf` to do forced includes of specified files or members. A forced include consists of a file or directory, data set, or data set member which will be parsed when a content assist operation is performed, regardless of whether that file or member was included in the source code using a pre-processor directive.

The file must be referenced in `zee.env` by the `include.c` or `include.cpp` variables before it is used. This reference in `zee.env` implies that you can specify a different file for usage by C and C++. The variables in `zee.env` are disabled by default.

The sample `include.conf` is located in `/etc/zexpl/`, unless you specified a different location when you customized and submitted job `FEL.SFELSAMP(FELSETUP)`. See “Customization setup” on page 13 for more details. You can edit the file with the TSO **OEDIT** command.

Definitions must start in column 1. Comment lines start with a pound sign (#) when using a US code page. Data lines can only have the name of a directory, file, data set or member. Comments are not allowed on the same line. Line continuations are not supported.

© Copyright IBM Corp. 2000, 2024

```
# To include the stdio.h file from the /usr/include directory, input:
# /usr/include/stdio.h
#
# To include all files of the /usr/include directory and all of it's
# sub-directories, input:
# /usr/include
#
# Uncomment and customize variable FILETYPES to limit the z/OS UNIX
# wildcard include to selected (case sensitive) file types:
# The file types are specified in a comma-delimited list (no blanks)
# FILETYPES=H,h,hpp,C,c,cpp,cxx

# To include all members of the CBC.SCLBH.H data set, input:
# //CBC.SCLBH.H
#
# To include the STDIOSTR member of the CBC.SCLBH.H  data set, input:
# //CBC.SCLBH.H(STDIOSTR)
# The sample list contains some commonly used C standard library files
/usr/include/assert.h
/usr/include/ctype.h
/usr/include/errno.h
/usr/include/float.h
/usr/include/limits.h
/usr/include/locale.h
/usr/include/math.h
/usr/include/setjmp.h
/usr/include/signal.h
/usr/include/stdarg.h
/usr/include/stddef.h
/usr/include/stdio.h
/usr/include/stdlib.h
/usr/include/string.h
/usr/include/time.h
```

*Figure 20. include.conf - Forced includes for C/C++ content assist*

# z/OS UNIX subprojects

This customization task does not require assistance, special resources, or special customization tasks.

---

REXEC (Remote Execution) is a TCP/IP service that enables clients to execute a command on the host system. SSH (Secure Shell) is a similar service, but all communication is encrypted. Developer for z/OS uses either service for doing remote (host-based) actions in z/OS UNIX subprojects.

**Notes:**

- Developer for z/OS uses the z/OS UNIX version of REXEC, not the TSO version.
- If REXEC/SSH is not configured to use the default port, the Developer for z/OS client must define the correct port for use by z/OS UNIX subprojects. This configuration can be done by selecting **Window** (on Windows) or **IBM Developer for z/OS** (on macOS) > **Preferences** > **z/OS Solutions** > **USS Subprojects** > **Remote Action Options** preference page. To know which port is used, see "REXEC or SSH setup" on page 62.

## REXEC or SSH setup

REXEC and SSH rely on services provided by INETD (Internet Daemon), which is another TCP/IP service. *Communications Server IP Configuration Guide* (SC31-8775) describes the steps required to set up INETD, REXEC, and SSH.

A common port used by REXEC is 512. To verify the port being used, check `/etc/inetd.conf` and `/etc/services`.

- Find the service name (1st word, `exec` in this example) of the `rexecd` server (7th word) in `/etc/inetd.conf`.

```
exec  stream tcp nowait OMVSKERN /usr/sbin/orexecd rexecd -LV
```

- Find the port (2nd word, 512 in this example) attached to this service name (1st word) in `/etc/services`.

```
exec      512/tcp      #REXEC   Command Server
```

The same principle applies to SSH. Its common port is 22, and the server name is `sshd`.

## Include preprocessor support

This customization task does not require assistance, special resources, or special customization tasks.

Developer for z/OS supports the interpreting and expanding COBOL and PL/I include statements, including select third-party include statements. Developer for z/OS also provides a sample REXX exec, FEKRNPLI, that can be called by the Developer for z/OS client to expand PL/I source by invoking the PL/I compiler.

FEKRNPLI is located in `FEL.#CUST.CNTL`, unless you specified a different location when you customized and submitted the `FEL.SFELSAMP(FELSETUP)` job. For more details, see Chapter 2, “Basic customization,” on page 13.

Customize the sample `FEL.#CUST.CNTL(FEKRNPLI)` exec, as described within the member. You must provide the following information:

- `compiler_hlq`: The high-level qualifier for the PL/I compiler

The Developer for z/OS client uses the TSO Command Service to execute the exec. This implies that if the FEKRNPLI exec is placed in the SYSPROC or SYSEXEC concatenation for the TSO Command Service, the user does not need to know the exact location of the exec. The user only needs to know the name. When using the Legacy ISPF Gateway, the SYSPROC or SYSEXEC concatenation is defined in `ISPF.conf`. Customization of this file is documented in *IBM Explorer for z/OS Host Configuration Guide*.

## ISPF EDIT macro support

This customization task does not require assistance, special resources, or special customization tasks.

Developer for z/OS supports executing ISPF EDIT macros against data sets and members, even when they are open in the Developer for z/OS editor. Developer for z/OS provides a sample REXX `exec`, FELEDTMC, that can be called by the Developer for z/OS client to invoke ISPF EDIT and execute a macro.

FELEDTMC is located in `FEL.#CUST.CNTL`, unless you specified a different location when you customized and submitted the `FEL.SFELSAMP(FELSETUP)` job. For more details, see Chapter 2, “Basic customization,” on page 13.

The sample FELEDTMC exec is fully functional, but it might need site specific adjustments.

Menu Manager in the client can query environment variables FEL_EDIT_MACRO_DSN and FEL_EDIT_MACRO to dynamically learn the location and name of the REXX exec it must invoke to use ISPF EDIT macro support.

## ISPF PACK support

This customization task does not require assistance, special resources, or special customization tasks.

When using the ISPF EDIT command **PACK ON | OFF**, ISPF uses a proprietary format to pack the data, and it does not provide a way to detect packed data without opening the member or data set first. This results in garbled data showing in the Developer for z/OS editor when a packed member or data set is opened. Developer for z/OS provides a REXX exec, **FELPACK**, that can be called by the client, for example in a Menu Manager action, to unpack or pack a member or data set.

**FELPACK** is located in `FEL.SFELPROC`, if you installed Developer for z/OS in the default location.

The Developer for z/OS client uses the TSO Command Service to execute the exec. This implies that if the **FELPACK** exec is placed in the SYSPROC or SYSEXEC concatenation for the TSO Command Service, the user does not need to know the exact location of the exec. The user only needs to know the name. When using the Legacy ISPF Gateway, the SYSPROC or SYSEXEC concatenation is defined in `ISPF.conf`. Customization of this file is documented in the *IBM® Explorer for z/OS® Host Configuration Guide*.

Menu Manager in the client can also query environment variable FEL_DSN_SFELPROC to dynamically learn the name of the SFELPROC data set, and thus call the FELPACK REXX exec directly.

## xUnit support for Enterprise COBOL and PL/I

This customization task does not require assistance, but does require the following resources or special customization tasks:

- PROCLIB update
- LINKLIST update

---

Frameworks that assist developers in writing code to perform repeatable, self-checking unit tests are collectively known as xUnit. Developer for z/OS provides such a framework for unit testing of Enterprise COBOL and PL/I code, called ZUnit.

**Note:** The ZUnit services documented here are superseded by the services offered by IBM z/OS Dynamic Test Runner FMID HAL6100. Ensure that this FMID is installed and configured, and use the steps described here only if FMID HAL6100 cannot be used. Let your users know which configuration is available on your host system. Keywords that they recognize are BZUPPLAY for IBM z/OS Dynamic Test Runner and AZUZUNIT for the configuration described here.

**Note:** If you want to use the AZU* test runner, you must install XML Toolkit 1.10 or later version.

To use the ZUnit framework, developers need access to the AZU* and IAZU* load modules in the `FEL.SFELLOAD` load library, either through STEPLIB or LINKLIST. The ZUnit test runner, AZUTSTRN, in turn needs access to various system libraries, either through STEPLIB or LINKLIST:

- `CEE.SCEERUN` and `CEE.SCEERUN2` (LE runtime)
- `SYS1.CSSLIB` (callable system services)
- `SYS1.SIXMLOD1` (XML toolkit)

The ZUnit test runner also needs access to a load library that holds the different test cases. This library is likely to be unique to a developer.

The ZUnit test runner, AZUTSTRN, can be called by the Developer for z/OS client in batch mode, from the TSO command line, and from the z/OS UNIX command line. When called in batch mode, the ZUnit test runner can be used to unit test programs that interact with Db2.

- Developer for z/OS provides sample procedures, AZUZUNIT and AZUZUDB2, to simplify the calling of the ZUnit test runner in batch mode. The sample procedures are located in `FEL.#CUST.PROCLIB`, unless you specified a different location when you customized and submitted the `FEL.SFELSAMP(FELSETUP)` job. For more details, see "Customization setup" on page 13.

  Customize the sample procedures, as described within the member, and copy them to `SYS1.PROCLIB`.

  The name of the procedure and the names of the steps in the procedure match the default properties that are included with the Developer for z/OS client. If the name of a procedure or the name of a step in a procedure is changed, the corresponding properties file on all of the clients must be updated. You should not change the procedure and step names.

  If the procedures cannot be copied into a system procedure library, ask the Developer for z/OS users to add a JCLLIB card right after the JOB card to their calling job.

```
//MYJOB    JOB <job parameters>
//PROCS    JCLLIB ORDER=(FEL.#CUST.PROCLIB)
```

- For calling the ZUnit test runner from z/OS UNIX (using the `/usr/lpp/IBM/zee/bin/zunit` script), you can specify the required non-LINKLIST data sets in the STEPLIB directive of `zee.env`, thus simplifying the setup for the developer.

  `zee.env` is located in `/etc/zexpl/`, unless you specified a different location when you customized and submitted `FEL.SFELSAMP(FELSETUP)` job. For more details, see "Customization setup" on page 13. You can edit the file with the TSO **OEDIT** command.

  The *zunit* script allows the user to specify data sets that will be added to the STEPLIB directive used by the script.
- For calling the ZUnit test runner from the TSO command line by using the `FEL.SFELPROC(FEKZUNIT)` exec, the system libraries must exist in LINKLIST. If they do not, developers must specify the system data set names on every call instance of the ZUnit test runner. You can also write a wrapper exec that does the **TSOLIB** allocations of these data sets for them. You can use FEKZUNIT itself as an example of how to code this wrapper exec.
- To start a debugger session at the test target source, EQAOPTS load module should be built and located in system search path like STEPLIB.

  **Note:** EQAOPTS load module data set needs to be put before the DT SEQAMOD data set in the search path. It should contain the following macro so that ZUnit would specify the name of target source:

```
EQAXOPT DLAYDBG,YES
EQAXOPT DLAYDBGDSN,'&&USERID.ZUNIT.DLAYDBG.EQAUOPTS
```

The ZUnit test runner allows for automatic reformatting of test reports. z/OS Explorer Extensions provides sample conversions (for example, conversion to Ant or jUnit format), which are located in `/usr/lpp/IBM/zee/samples/zunit/xsd` and `/usr/lpp/IBM/zee/samples/zunit/xsl`, if you installed Developer for z/OS in the default `/usr/lpp/IBM/zee` location.

## xUnit support for CICS applications (ZUnit)

You need the assistance of a CICS , TCP/IP and possibly a security administrator to complete this customization task, which requires specific resources or special customization tasks.

- Update CICS region JCL
- Define resources to CICS
- Reserve a TCP/IP port
- Define security profiles and permits

---

Frameworks that assist developers in writing code to perform repeatable, self-checking unit tests are collectively known as xUnit. IBM Developer for z/OS provides such a framework for unit testing of Enterprise COBOL and PL/I code, called ZUnit. This customization task extends ZUnit to support testing CICS applications through recording of the parameters used in EXEC CICS calls.

**Note:** Support for testing CICS applications relies on services offered by IBM z/OS Dynamic Test Runner, FMID HAL6xxx. Ensure that this FMID is installed and configured. You might find it convenient to perform the customization tasks described here in conjunction with those described in Support for CICS applications of the Dynamic Test Runner Host Configuration Guide as there are common resources updated by both tasks.

To record a CICS application, ZUnit requires the following CICS updates:

- CICS system initialization (SIT) parameter updates
  - Specify `TCPIP=YES`.
  - Specify `USSHOME=/usr/lpp/cicsts/cicsts61`, where `/usr/lpp/cicsts/cicsts61` is the path used during the SMP/E installation of CICS Transaction Server.

  **Note:** Security considerations documents the impact of specifying these optional SIT parameters:

  - SEC=YES
  - CMDSEC=ALWAYS
  - RESSEC=ALWAYS
- CICS JCL updates:
  - Specify `REGION=0M` on the EXEC statement to provide sufficient room to buffer the recording.
  - Define the `FEL.SFELLOAD` load library in the region's `DFHRPL DD` statement, where `FEL` is the high level qualifier used during the SMP/E install of z/OS Explorer Extensions.
    - Alternatively, you can define a `LIBRARY` resource definition in the CSD for this load library. A sample definition can be found in the `FEL.SFELSAMP(AZUCSD)` sample CSD update job.
- CICS CSD updates:
  - Define ZUnit to your CICS region, as documented in the AZUCSD sample CSD update job. AZUCSD is located in FEL.#CUST.JCL, unless you specified a different location when you customized and submitted job FEL.SFELSAMP(FELSETUP). For more information. see "Customization setup" on page 13.

The ZUnit CICS recorder uses a CICS PIPELINE to communicate with the ZUnit client. This CICS service requires that a TCP/IP port be defined in the CSD. The developers who use this service need to know the host system name and the port on which the service is listening.

The ZUnit CICS recorder uses a temporary storage queue in auxiliary storage to store log data. The log data is written to a queue named `_AZU_LOGRS_`. You can browse the log data with the CICS command CEBR. When the log data is too big to reserve TS queues in auxiliary storage for other programs, purge it manually in the CEBR command screen. Abend `AIER (resp-18 NOSPACE)` occurs when an auxiliary TS queue runs out of space.

You can run the **AZUM** transaction from a CICS terminal to manage logging. When the log level is set to a larger value, more data is output to the log files. The syntax of the **AZUM** transaction is:

```
AZUM { GETLOGLEVEL | SETLOGLEVEL0 | SETLOGLEVEL1 | SETLOGLEVEL2 }
```

**GETLOGLEVEL**
Show the current log level

**SETLOGLEVEL0**
Set the current log level to 0 (error only).

**SETLOGLEVEL1**
Set the current log level to 1 (error and warning)

**SETLOGLEVEL2**
Set the current log level to 2 (error, warning and information)

## Security considerations

Multiple user IDs are involved when using ZUnit to test CICS applications:

1. The user ID used by the end-user when connecting to the host using the IBM Developer for z/OS client, referred to as the *client user ID* in this section.
2. The user ID used by the CICS region when doing actions requested by ZUnit, referred to as the *CICS user ID* in this section. The CICS user ID can differ from the client user ID, for example it can be the CICS default user.
3. The user ID used by an administrator or end-user to interact directly with CICS, referred to as *CICS admin ID* in this section. The CICS admin ID can differ from both the client user ID and the CICS user ID.
4. The user ID that the CICS region runs under, referred to as the *CICS region user ID* in this section.

**Data set**

As part of its processing, ZUnit creates a recording (also known as "playback") data set that will be populated by Dynamic Test Runner. The high level qualifier(s) of this data set can be configured in the ZUnit client preferences (default is the client user ID followed by ZUNIT.PB). The low level qualifier is the name of the program under test. Two user IDs require authorization to the security profile in the DATASET class that protects these recording data sets:

- The client user ID requires ALTER access so that it can create them
- The CICS region user ID requires UPDATE access so that it can write to them

For example, use the following z/OS Security Server (RACF) commands to define a data set profile for client user ID IBMUSER using the default high level qualifier, and to authorize users to use it:

```
ADDSD 'IBMUSER.ZUNIT.PB.*' UACC(NONE) DATA('ZUNIT PLAYBACK')
PERMIT 'IBMUSER.ZUNIT.PB.* ' CLASS(DATASET) ACCESS(ALTER) ID(IBMUSER)
PERMIT 'IBMUSER.ZUNIT.PB.* ' CLASS(DATASET) ACCESS(UPDATE) ID(#CICS_region_user_ID)
SETROPTS GENERIC(DATASET) REFRESH
```

**SEC=YES**

When CICS security is enabled with the SEC=YES system initialization parameter, users of the ZUnit management transaction (CICS admin ID) require READ access to the profile protecting the AZUMCICS transaction. Default resource and grouping class names for transactions are TCICSTRN and GCICSTRN, respectively, but these names can be overridden by the XTRAN CICS system initialization parameter.

For example, use the following z/OS Security Server (RACF) commands to define a profile for transaction AZUM in the TCICSTRN class, and to authorize users to run it:

```
RDEFINE TCICSTRN AZUM UACC(NONE)DATA('ZUNIT MANAGEMENT')
PERMIT AZUM CLASS(TCICSTRN) ACCESS(READ) ID(#CICS_admin_ID_group)
SETROPTS GENERIC(TCICSTRN) REFRESH
```

No further security configuration is required unless CICS security is enabled with the SEC=YES system initialization parameter, and either CMDSEC=ALWAYS or RESSEC=ALWAYS is specified as a CICS system initialization parameter.

**CMDSEC=ALWAYS**

If SEC=YES and CMDSEC=ALWAYS are specified as CICS system initialization parameters, then client user IDs require ALTER access to TDQUEUE, the security profile that protects the CICS Transient Data queue (TD queue) commands. Default resource and grouping class names for CICS commands are CCICSCMD and VCICSCMD, respectively, but these names can be overridden by the XCMD CICS system initialization parameter.

For example, use the following z/OS Security Server (RACF) commands to define a profile for TD queue commands in the CICSCMD class, and to authorize users to issue the TDQUEUE commands:

```
RDEFINE CCICSCMD TDQUEUE UACC(NONE)
PERMIT TDQUEUE CLASS(CCICSCMD) ACCESS(ALTER) ID(#client_user_ID_group)
SETROPTS GENERIC(CCICSCMD) REFRESH
```

**RESSEC=ALWAYS**

If SEC=YES and RESSEC=ALWAYS are specified as CICS system initialization parameters, then client user IDs require the following resource authorization:

**Transient data queues (TD queues)**

Client user IDs require ALTER access to ZU*, the profile that protects ZUnit CICS TD queues. Default resource and grouping class names for CICS TD queues are DCICSDCT and ECICSDCT, respectively, but these names can be overridden by the XDCT CICS system initialization parameter.

For example, use the following z/OS Security Server (RACF) commands to define a profile for TD queues in the DCICSDCT class, and to authorize users to access them:

```
RDEFINE DCICSDCT ZU* UACC(NONE) DATA('ZUNIT')
PERMIT ZU* CLASS(DCICSDCT) ACCESS(ALTER) ID(#client_user_ID_group)
SETROPTS GENERIC(DCICSDCT) REFRESH
```

**Programs**

Client user IDs require:

- READ access to AZUCREST
- READ access to all load modules or program objects containing routines that are called by programs under test using non-CICS interfaces

Default resource and grouping class names for CICS programs are MCICSPPT and NCICSPPT, respectively, but these names can be overridden by the XPPT CICS system initialization parameter.

For example, use the following z/OS Security Server (RACF) commands to define profiles for program AZUCREST in the MCICSPPT class and programs under test in the NCICSPPT class, and to authorize users to access them:

```
RDEFINE MCICSPPT AZUCREST UACC(NONE) DATA('ZUNIT')
RDEFINE NCICSPPT TESTPGM1 UACC(NONE)
ADDMEM(#test_program_1,#test_program_2,#test_program_3)
RDEFINE NCICSPPT TESTPGM2 UACC(NONE) ADDMEM(#test_program_4,#test_program_5)
PERMIT AZUCREST CLASS(MCICSPPT) ACCESS(READ) ID(#client_user_ID_group)
PERMIT TESTPGM1 CLASS(NCICSPPT) ACCESS(READ) ID(#client_user_ID_group1)
PERMIT TESTPGM2 CLASS(NCICSPPT) ACCESS(READ) ID(#client_user_ID_group2)
SETROPTS GENERIC(MCICSPPT) REFRESH
SETROPTS GENERIC(NCICSPPT) REFRESH
```

**Temporary Storage queues (TS queues)**

By default, no authorization is required for temporary storage queues (TS queues). If you want to enforce security for TS queues, define two TSMODELs in CICS:

- One with PREFIX(_AZU_LOG) LOCATION(AUXILIARY) SECURITY(YES)
- One with PREFIX(_AZU_CTL) LOCATION(MAIN) SECURITY(YES)

Client user IDs require UPDATE authorization to, at least, profiles for TS queues with names starting with _AZU_LOGRS_ and _AZU_CTL2_.

The default resource and grouping class names for temporary storage queues are SCICSTST and UCICSTST, respectively, but these names can be overridden by the XTST CICS system initialization parameter.

For example, use the following z/OS Security Server (RACF) commands to define profiles for TS queues used by ZUnit, and to authorize users to access them:

```
RDEFINE SCICSTST _AZU_* UACC(NONE) DATA('ZUNIT')
PERMIT _AZU_* CLASS(SCICSTST) ACCESS(UPDATE) ID(#client_user_ID_group)
SETROPTS GENERIC(SCICSTST) REFRESH
```

## Limitations

The ZUnit CICS recorder has the following limitations.

- Only 31-bit load modules are supported.
- The maximum supported length of one parameter in a CICS command is 262032 bytes. If it goes beyond 262032 bytes, only the first 262032 bytes are recorded.
- If the recorded program is being called through a static program, program calls will not be captured in the recording.

# Enterprise Service Tools support

This customization task does not require assistance, special resources, or special customization tasks.

The Developer for z/OS client has a code generation component called Enterprise Service Tools. Depending on the type of code being generated, this code relies on functions provided by the Developer for z/OS host system installation. Making these host system functions available is described in the following sections:


- "CICS bidirectional language support" on page 69
- "Diagnostic IRZ messages for Enterprise Service Tools" on page 70


# CICS bidirectional language support

You need the assistance of a CICS administrator to complete this customization task, which requires the following resources or special customization tasks:

- Update CICS region JCL
- Define a program to CICS

The Developer for z/OS Enterprise Service Tools component supports different formats of Arabic and Hebrew interface messages, and bidirectional data presentation and editing in all editors and views. In terminal applications, both left-to-right and right-to-left screens are supported, and numeric fields and fields with opposite-to-screen orientation.

Additional bidirectional features and functionality include the following:

- The Enterprise Service Tools service requestor dynamically specifies bidirectional attributes of interface messages.
- Enterprise Service Tools-generated runtime code includes conversion of data between fields in messages that have different bidirectional attributes.

Additionally, Enterprise Service Tools-generated code can support bidi transformation in environments such as batch applications. You can make the Enterprise Service Tools generators to include calls to the bidirectional conversion routines by specifying the appropriate bidi transformation options in the Enterprise Service Tools generation wizards and linking the generated programs with the appropriate bidirectional conversion library, FEL.SFELLOAD.

To activate CICS Bidirectional language support, perform the following tasks:

1. Place the FEL.SFELLOAD load modules FEJBDCMP and FEJBDTRX in the CICS RPL concatenation (DD statement DFHRPL). You should do this by adding the installation data set to the concatenation so that applied maintenance is automatically available to CICS.

   **Important:** If you do not concatenate the installation data set but copy the modules into a new or existing data set, keep in mind that those modules are DLLs and must reside in a PDSE library.
2. Define FEJBDCMP and FEJBDTRX as programs to CICS by using the appropriate CEDA command.

```
CEDA DEF PROG(FEJBDCMP) LANG(LE) G(xxx)
CEDA DEF PROG(FEJBDTRX) LANG(LE) G(xxx)
```

# Diagnostic IRZ messages for Enterprise Service Tools

This customization task does not require assistance, but does require the following resources or special customization tasks:

- LINKLIST update
- CICS region JCL update
- CICS region CSD update

---

The Developer for z/OS client has a code generation component called Enterprise Service Tools. For the code generated by Enterprise Service Tools to issue diagnostic error messages, all IRZM* and IIRZ* modules in the FEL.SFELLMOD load library must be made available to the generated code. Enterprise Service Tools can generate code for the following environments:

- CICS
- IMS
- MVS batch

Use the following instructions when the generated code is executed in a CICS transaction.

- Add all IRZM* and IIRZ* modules in FEL.SFELLMOD to the DFHRPL DD of the CICS region. You should do this by adding the installation data set to the concatenation so that applied maintenance is automatically available.
- Customize and submit the IRZCSD job to update the CICS System Definition (CSD) for the CICS region. For customization instructions, see the documentation within the member. IRZCSD is located in FEL.#CUST.JCL, unless you specified a different location when you customized and submitted the FEL.SFELSAMP(FELSETUP) job. For more details, see "Customization setup" on page 13.

In all other situations, make all IRZM* and IIRZ* modules in FEL.SFELLMOD available either through STEPLIB or LINKLIST. You should do this by adding the installation data set to the concatenation so that applied maintenance is automatically available.

If you use STEPLIB, define the modules not available through LINKLIST in the STEPLIB directive of the task that executes the code.

If the load modules are not available and the generated code encounters an error, the following message is issued:

```
IRZ9999S Failed to retrieve the text of a Language Environment runtime
message. Check that the Language Environment runtime message module for
facility IRZ is installed in DFHRPL or STEPLIB.
```

**Note:**

- Module FEL.SFELLMOD(IRZPWSIO) is statically linked during top-down IMS MPP code generation. Therefore, the module must not be available during run time of the generated code. It should be available only during compile time.
- In version 9.0.1, FEL.SFELLMOD(IRZPWSIO) and the related FEL.SFELSAMP(IRZPWSH) sample PL/I include member moved from Developer for z/OS to IMS Version 12. The parts are renamed to IMS.SDFSRESL(DFSPWSIO) and IMS.SDFSSMPL(DFSPWSH) respectively.

# Chapter 6. Installation verification

After completing the product customization, you can use the Installation Verification Programs (IVPs) described in this chapter to verify the successful setup of key product components.

## Verify the services

The Developer for z/OS installation provides several Installation Verification Programs (IVPs) for the basic and optional services. The IVP scripts are located in the installation directory which, by default, is `/usr/lpp/IBM/zee/bin/`.

*Table 13. IVPs for services*

| Script name | Description |
|---|---|
| felfivpc | "CARMA connection" on page 72 |

The tasks described in the following sections require you to be active in z/OS UNIX. This can be done by issuing the **OMVS** TSO command. To return to TSO, use the **exit** command.

A large region size is required for the user ID that executes the IVPs because functions such as Java, which require a lot of memory, are executed. You should set the region size to 131072 kilobytes (128 megabytes) or more.

The following sample error is a clear indication of an insufficient region size, but other errors can occur, too. For example, Java might fail to start.

```
CEE5213S The signal SIGPIPE was received.
%z/OS UNIX command%: command was killed by signal number 13
    %line-number% *-*   %REXX command%
       +++ RC(137) +++
```

**Note:** The z/OS Explorer and Developer for z/OS started tasks must be active before starting the IVP test.

## IVP initialization

All sample commands in this section require certain environment variables to be set. This way, the IVP scripts are available through the PATH statement and the location of the customized configuration files is known. Use the **pwd** and **cd** commands to verify and change your current directory to the directory with the customized configuration files. The `ivpinit` shell script can then be used to set the RSE environment variables, such as in the following sample, where $ is the z/OS UNIX prompt:

```
$ pwd
/u/userid
$ cd /etc/zexpl
$ . ./ivpinit
-- RSE_CFG set to /etc/zexpl -- based on current location
-- RSE_HOME set to /usr/lpp/IBM/zexpl -- defined in $RSE_CFG/rse.env
-- added product $PATH to PATH
-- PATH=/etc/zexpl:/bin:/usr/lpp/java/J8.0/bin:/usr/lpp/IBM/zexpl/bin:/usr/lpp/i
spf/bin:/usr/lpp/IBM/zee/bin:/bin
```

The first period (.) in `. ./ivpinit` is a z/OS UNIX command to run the shell in the current environment, so that the environment variables set in the shell are effective even after exiting the shell. The second period (.) is referring to the current directory.

**Note:**

© Copyright IBM Corp. 2000, 2024

- If `. ./ivpinit` is not executed before the `felfivp*` scripts, the path to these scripts must be specified when calling them, as in the following sample:

  ```
  /usr/lpp/IBM/zee/bin/felfivpc
  ```

  Also, if `. ./ivpinit` is not executed first, all `felfivp*` scripts ask for the location of the directory holding the customized configuration files.

## CARMA connection

Verify the connection to CARMA by executing the following command:

```
felfivpc
```

The command should return a success message.:

**Note:** If the IVP fails, verify the content of `/tmp/felfivpc.log`. This log documents the communication between RSE and CARMA and might contain information that helps to find the root cause of the failure.

`felfivpc` has the following optional, non-positional, parameters:

**-noram**
By default, `felfivpc` starts the first RAM that is defined in the CRADEF VSAM data set. There might be instances when you do not want to test the RAM; for example, a third-party RAM is listed first, and it requires unexpected input. In such cases, you can use the `–noram` startup argument to omit the RAM-specific steps of the IVP test.

# Chapter 7. Security definitions

Customize and submit the sample FELRACF job, which has sample RACF commands to create the basic security definitions for Developer for z/OS.

FELRACF is located in FEL.#CUST.JCL, unless you specified a different location when you customized and submitted the FEL.SFELSAMP(FELSETUP) job.

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

## Requirements and checklist

To complete the security setup, the security administrator must know the values that are listed in Table 14 on page 73. These values were defined during previous steps of the installation and customization of Developer for z/OS.

*Table 14. Security setup variables*

| **Description** | • **Default value**<br>• **Where to find the answer** | **Value** |
|---|---|---|
| Developer for z/OS product high-level qualifier | • FEL<br>• SMP/E installation | |
| Developer for z/OS customization high-level qualifier | • FEL.#CUST<br>• FEL.SFELSAMP(FELSETUP), as described in "Customization setup" on page 13. | |

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

- "Define the data set profiles" on page 73
- "Verify the security settings" on page 74

## Define the data set profiles

READ access for users and ALTER for system programmers is sufficient for most Developer for z/OS data sets. Replace the #sysprog placeholder with valid user IDs or RACF group names. Also, ask the system programmer who installed and configured the product for the correct data set names. FEL is the default high-level qualifier used during installation and FEL.#CUST is the default high-level qualifier for data sets created during the customization process.

- ```
  ADDGROUP (FEL) OWNER(IBMUSER) SUPGROUP(SYS1)
  DATA('IBM Developer for z/OS - HLQ STUB')
  ```
- ```
  ADDSD  'FEL.*.**' UACC(READ)
  DATA('IBM Developer for z/OS')
  ```
- ```
  PERMIT 'FEL.*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
  ```
- ```
  SETROPTS GENERIC(DATASET) REFRESH
  ```

© Copyright IBM Corp. 2000, 2024

**Notes:**

- Protect FEL.SFELLPA against updates because this data set is loaded into LPA, which is APF authorized by default.
- The sample commands in this publication and in the FELRACF job assume that Enhanced Generic Naming (EGN) is active. When EGN is active, the ** qualifier can be used to represent any number of qualifiers in the DATASET class. Substitute ** with * if EGN is not active on your system. For more information about EGN, see *Security Server RACF Security Administrator's Guide* (SA22-7683).

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

- CARMA RAM (Repository Access Manager) developers require UPDATE access to the CARMA VSAMs, FEL.#CUST.CRA*.
  - ```
    ADDSD  'FEL.#CUST.CRA*.**' UACC(READ)
    DATA('IBM Developer for z/OS - CARMA')
    ```
  - ```
    PERMIT 'FEL.#CUST.CRA*.**' CLASS(DATASET) ACCESS(ALTER) ID(#sysprog)
    ```
  - ```
    PERMIT 'FEL.#CUST.CRA*.**' CLASS(DATASET) ACCESS(UPDATE) ID(#ram-developer)
    ```
  - ```
    SETROPTS GENERIC(DATASET) REFRESH
    ```

## Verify the security settings

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

- Data set profiles
  - `LISTGRP FEL`
  - `LISTDSD PREFIX(FEL) ALL`

# Chapter 8. Migration guide

## Migration considerations

This section highlights installation and configuration changes compared to previous releases of the product. It also gives some general guidelines to migrate to this release. For more information, see the related sections in this manual.

- If you are a previous user of IBM Developer for z/OS, save the related customized files before upgrading to this version of IBM Developer for z/OS.
- If you plan on running multiple instances of Developer for z/OS, read "Running multiple instances" in the *IBM Explorer for z/OS Host Configuration Reference*.
- If your migration scenario spans more than two releases, you should do the customizations again, as if there is no older release present.

### Backing up the previously configured files

If you are a previous user of Developer for z/OS, save the related customized files before installing this version of IBM Developer for z/OS.

Customizable Developer for z/OS files can be found at the following locations:

- Older versions up to version 9.5.0
  - `FEK.SFEKSAMP`, some members are copied to `FEK.#CUST.*` by the FEKSETUP sample job, where * equals `PARMLIB`, `PROCLIB`, `JCL`, `CNTL`, `ASM` and `COBOL`
  - `FEK.SFEKSAMV`
  - `/usr/lpp/rdz/samples/`, some files are copied to `/etc/rdz/` and `/var/zexpl/sclmdt/*` by the FEKSETUP sample job, where * equals `CONFIG/`, `CONFIG/PROJECT/` and `CONFIG/script/`
- Version 9.5.1
  - `FEL.SFELSAMP`, some members are copied to `FEL.#CUST.*` by the FELSETUP sample job, where * equals `PARMLIB`, `PROCLIB`, `JCL`, `CNTL`, `ASM` and `SQL`
  - `FEL.SFELSAMV`
  - `/usr/lpp/IBM/rdz/samples/`, some files are copied to `/etc/zexpl/` and `/var/rdz/sclmdt/*` by the FELSETUP sample job, where * equals `CONFIG/`, `CONFIG/PROJECT/` and `CONFIG/script/`
- Version 14.0 and 14.1
  - `FEL.SFELSAMP`, some members are copied to `FEL.#CUST.*` by the FELSETUP sample job, where * equals `PARMLIB`, `PROCLIB`, `JCL`, `CNTL`, `ASM` and `SQL`
  - `FEL.SFELSAMV`
  - `/usr/lpp/IBM/idz/samples/`, some files are copied to `/etc/zexpl/` and `/var/idz/sclmdt/*` by the FELSETUP sample job, where * equals `CONFIG/`, `CONFIG/PROJECT/` and `CONFIG/script/`
- Version 14.2
  - `FEL.SFELSAMP`, some members are copied to `FEL.#CUST.*` by the FELSETUP sample job, where * equals `PARMLIB`, `PROCLIB`, `JCL`, `CNTL`, `ASM` and `SQL`
  - `/usr/lpp/IBM/zee/samples/`, some files are copied to `/etc/zexpl/` by the FELSETUP sample job

© Copyright IBM Corp. 2000, 2024

# Migrating to Enterprise Edition

To migrate an IBM Developer for z/OS server from the standard edition to Enterprise Edition or to Application Delivery Foundation for z/OS, you must update several host configuration files.

1. Update the product registration record in SYS1.PARMLIB(IFAPRDxx).

   Products to be enabled on z/OS are defined in SYS1.PARMLIB(IFAPRDxx). Define PROD=xx in the IEASYSxx parmlib member to specify which IFAPRDxx parmlib member should be used during IPL.

   Specify the following in IFAPRDxx to define IBM Developer for z/OS Enterprise Edition (product code 5755-AC5):

   ```
   PRODUCT OWNER('IBM CORP')
           NAME('IBM IDz EE')
           ID(5755-AC5)
           VERSION(*) RELEASE(*) MOD(*)
           FEATURENAME(*)
           STATE(ENABLED)
   ```

   Specify the following in IFAPRDxx to define IBM Application Delivery Foundation for z/OS (product code 5655-AC6):

   ```
   PRODUCT OWNER('IBM CORP')
           NAME('IBM APP DLIV FND')
           ID(5655-AC6)
           VERSION(*) RELEASE(*) MOD(*)
           FEATURENAME(*)
           STATE(ENABLED)
   ```

   After the IFAPRDxx parmlib member is updated, it can be activated dynamically (until the next IPL) with the following console command:

   ```
   SET PROD=xx
   ```

   **Note:** All product flavors of the host components register the following features:

   - FEL-RSED (for RSE daemon in z/OS Explorer Extensions, FMID HHOPxxx)
   - AKG-CC (for Code Coverage in z/OS Source Code Analysis, FMID HAKGxxx)
   - AKG-CR (for Code Review in z/OS Source Code Analysis, FMID HAKGxxx)

2. If not yet done, update SYS1.PARMLIB(SMFPRMxx) to enable SMF record collection.

   System Management Facilities (SMF) uses SYS1.PARMLIB(SMFPRMxx) to determine which record types and subtypes should be collected. Define SMF=xx in the IEASYSxx parmlib member to specify which SMFPRMxx parmlib member should be used during IPL.

   Developer for z/OS creates SMF records type 122, subtype 1. Collection of these records is required if you want to use the Developer for z/OS Enterprise Edition host to activate all Developer for z/OS client functions.

   Update the SYS(TYPE()) definition in SMFPRMxx to collect type 122 records.

   Updated SMFPRMxx definitions can be activated dynamically (until the next IPL) with the following console command:

   ```
   SET SMF=xx
   ```

3. IBM Developer for z/OS Enterprise Edition and IBM Application Delivery Foundation for z/OS include additional FMIDs, such as IBM Dependency Based Build. To obtain the additional FMIDs, order the product you want to migrate to in Shopz. For installation instructions for other host software products, see Host Program Directories.

4. Enable auditing if you need or want to track user logon actions for IBM Developer for z/OS Enterprise Edition. For more information about audit logging, see Audit logging in the IBM Explorer for z/OS documentation. You can also write an exit routine for the logon.action exit point. For more information about user exits, see User exit considerations.

5. For other migration considerations and tools, see the related links.

**Related information**

Product enablement in IFAPRDxx
SMF record collection in SMFPRMxx
IBM Developer for z/OS client-server compatibility
Host Configuration Assistant for Z Development
Managing licenses

# Migrate from version 14.2 to version 15.0

These notes are for a migration from a base version 14.2 to version 15.0. It includes changes that are already documented as part of version 14.2 maintenance. The changes that are part of the maintenance stream, and thus possibly already implemented, are marked with the release where they were introduced.

## IBM z/OS Explorer Extensions, FMID HHOPF00

- The default SMP/E install location for MVS components did not change and remains FEL.*.
- The default SMP/E install location for z/OS UNIX components did not change and remains /usr/lpp/IBM/zee/*.
- Customization: The following members are no longer provided:
  - FELCMPLY (SFELSAMP / #CUST.JCL)
- Since PTF UI69420:
  - Proclib: Customizable members have been updated:
    - PROCLIB(ELAXF): SET BZU
    - PROCLIB(AZUZUNIT): SBZULOAD in STEPLIB
- Since PTF UI68285:
  - RSE: A customizable file has been updated:
    - zee.env: DEFAULT_REMOTE_ZOS_FILE_SEARCH
- Since PTF UI66750:
  - ZUnit: A customizable member has been updated:
    - JCL(AZUCSD): replace content to interact with FMID HAL6xxx

### Configurable files

Table 15 on page 77 shows an overview of z/OS Explorer Extensions files that are customized in version 15.0. The z/OS Explorer Extensions sample libraries, FEL.SFELSAMP and /usr/lpp/IBM/zee/samples/, contain more customizable members than listed here, such as sample CARMA source code and jobs to compile them.

Note: Sample job FELSETUP copies all listed members to different data sets and directories, default FEL.#CUST.* and /etc/zexpl/*.

*Table 15. Version 15.0 customizations*

| Member/File | Default location | Purpose | Migration notes |
|---|---|---|---|
| FELSETUP | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL to create data sets and directories, and populate them with customizable files | Updated to remove obsolete members |

*Table 15. Version 15.0 customizations (continued)*

| Member/File | Default location | Purpose | Migration notes |
|---|---|---|---|
| ELAXF | `FEL.SFELSAMP [FEL.#CUST.PROCLIB]` | JCL INCLUDE to manage HLQs | Updated to add new definitions |
| ELAXF* | `FEL.SFELSAMP [FEL.#CUST.PROCLIB]` | JCL for remote project builds, and so on | None |
| FELRACF | `FEL.SFELSAMP [FEL.#CUST.JCL]` | JCL for security definitions | None |
| FELSMF | `FEL.SFELSAMP[ FEL.#CUST.JCL]` | JCL forSMF Reporting | None |
| FELSMF | `FEL.SFELSAMP [FEL.#CUST.JCL]` | JCL for SMF reporting | Update to call sample REXX |
| CRA$VMSG | `FEL.SFELSAMP [FEL.#CUST.JCL]` | JCL to create the CARMA message VSAM | None |
| CRA$VDEF | `FEL.SFELSAMP [FEL.#CUST.JCL]` | JCL to create the CARMA configuration VSAM | None |
| CRA$VSTR | `FEL.SFELSAMP [FEL.#CUST.JCL]` | JCL to create the CARMA custom information VSAM | None |
| CRA$VCAD | `FEL.SFELSAMP [FEL.#CUST.JCL]` | JCL to create the CARMA configuration VSAM for CA Endevor® SCM RAM | None |
| CRA$VCAS | `FEL.SFELSAMP [FEL.#CUST.JCL]` | JCL to create the CARMA custom information VSAM for CA Endevor® SCM RAM | None |
| CRASUBMT | `FEL.SFELSAMP [FEL.#CUST.CNTL]` | CARMA batch startup CLIST | None |
| CRASUBCA | `FEL.SFELSAMP [FEL.#CUST.CNTL]` | CARMA batch startup CLIST for CA Endevor® SCM RAM | None |
| CRACFG | `FEL.SFELSAMP [FEL.#CUST.PARMLIB]` | CARMA interaction configuration for CA Endevor® SCM RAM | None |

| *Table 15. Version 15.0 customizations (continued)* | | | |
|---|---|---|---|
| **Member/File** | **Default location** | **Purpose** | **Migration notes** |
| CRABCFG | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA batch actions configuration for CA Endevor® SCM RAM | None |
| CRABATCA | FEL.SFELSAMP [FEL.#CUST.CNTL] | CARMA batch action JCL for CA Endevor® SCM RAM | None |
| CRASCL | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | Template SCL for CA Endevor® SCM | None |
| CRASHOW | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA configuration for CA Endevor® SCM RAM | None |
| CRATMAP | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA configuration for CA Endevor® SCM RAM | None |
| CRANDVRA | FEL.SFELPROC | CARMA allocation REXX for CA Endevor® SCM RAM | None |
| CRADYNDA | FEL.SFELPROC | CARMA allocation REXX for CA Endevor® SCM RAM | None |
| CRAALLOC | FEL.SFELPROC | CARMA allocation REXX | None |
| CRA#VSLM | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the SCLM RAM's message VSAM | None |
| CRA#ASLM | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the SCLM RAM's data sets | None |
| CRA#VPDS | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the PDS RAM's message VSAM | None |
| CRA#UADD | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to merge RAM definitions | None |
| CRA#UQRY | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to extract RAM definitions | None |

| *Table 15. Version 15.0 customizations (continued)* | | | |
|---|---|---|---|
| **Member/File** | **Default location** | **Purpose** | **Migration notes** |
| CRAXJCL | FEL.SFELSAMP [FEL.#CUST.ASM] | Sample source code for IRXJCL replacement | None |
| CRA#CIRX | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to compile CRAXJCL | None |
| FELTEP2 | FEL.SFELSAMP [FEL.#CUST.SQL] | SQL command input used by ELAXF* | None |
| FELTIAD | FEL.SFELSAMP [FEL.#CUST.SQL] | SQL command input used by ELAXF* | None |
| AZUALLOC | FEL.SEFLSAMP [FEL.#CUST.JCL] | JCL for ZUnit dataset allocation | None |
| AZUCSD | FEL.SEFLSAMP [FEL.#CUST.JCL] | JCL to define ZUnit to CICS | Updated to utilize FMID HAL6100 |
| AZUZUDB2 | FEL.SFELSAMP [FEL.#CUST.PROCLIB] | JCL procedure for ZUnit | None |
| AZUZUNIT | FEL.SFELSAMP [FEL.#CUST.PROCLIB] | JCL procedure for ZUnit | SBZULOAD in STEPLIB |
| FEKRNPLI | FEL.SFELSAMP [FEL.#CUST.CNTL] | REXX to call the PL/I compiler from within the preprocessor framework | None |
| IRZCSD | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to define Enterprise Service Tools to CICS | None |
| zee.env | /usr/lpp/IBM/zee/samples/ [/etc/zexpl/] | RSE environment variables | Add DEFAULT_REMOTE_ZOS_FILE_SEARCH |
| CRASRV.properties | /usr/lpp/IBM/zee/samples/ [/etc/zexpl/] | CARMA configuration file | None |
| crastart.conf | /usr/lpp/IBM/zee/samples/ [/etc/zexpl/] | CARMA configuration file for CRASTART usage | None |

| Table 15. Version 15.0 customizations (continued) | | | |
|---|---|---|---|
| **Member/File** | **Default location** | **Purpose** | **Migration notes** |
| crastart.endevor.conf | /usr/lpp/IBM/zee/samples/[/etc/zexpl/] | CARMA configuration file for CRASTART usage for CA Endevor® SCM RAM | None |
| include.conf | /usr/lpp/IBM/zee/samples/[/etc/zexpl/] | Forced includes for C/C++ content assist | None |

## IBM z/OS Source Code Analysis, FMID HAKGF00

- The default SMP/E install location for MVS did not change and remains AKG.*.
- The default SMP/E install location for z/OS UNIX did not change and remains /usr/lpp/IBM/akg/*.
- The z/OS UNIX components of Code Coverage moved to the z/OS Debugger FMID, HADRF00.

### Configurable files

Table 16 on page 81 shows an overview of files that are customized in version 15.0. The IBM z/OS Source Code Analysis sample libraries, AKG.SAKGSAMP and /usr/lpp/IBM/akg/samples, contain more customizable members than listed here, such as sample code review post-processing script.

**Note:** Sample job AKGSETUP copies all listed members to different data sets, default AKG.#CUST.*.

| Table 16. Version 15.0 customizations | | | |
|---|---|---|---|
| **Member or File** | **Default location** | **Purpose** | **Migration notes** |
| AKGSETUP | AKG.SAKGSAMP [AKG.#CUST.JCL] | JCL to create data sets, and populate them with customizable files | None |
| AKGCC | AKG.SAKGSAMP [AKG.#CUST.PROCLIB] | JCL for code coverage | Update default path names |
| AKGCR | AKG.SAKGSAMP [AKG.#CUST.PROCLIB] | JCL for code review | None |
| AKGCRADD | AKG.SAKGSAMP [AKG.#CUST.JCL] | JCL to add third-party code to code review | None |

# Migrate from version 14.1 to version 14.2

These notes are for a migration from a base version 14.1 to version 14.2. It includes changes that are already documented as part of version 14.1 maintenance. The changes that are part of the maintenance stream, and thus possibly already implemented, are marked with the release where they were introduced.

## IBM z/OS Explorer Extensions, FMID HHOPE20

- The FMID is renamed from IBM Developer for z Systems to IBM z/OS Explorer Extensions.

- SCLM Developer Toolkit, which was deprecated in a previous release, is no longer shipped.
- The default SMP/E install location for MVS components did not change and remain FEL.*.
- The default SMP/E install location for z/OS UNIX components changed from /usr/lpp/IBM/idz/* to /usr/lpp/IBM/zee/*.
- Customization: The following path is no longer used:
  - /var/idz
- Customization: Customizable files have been updated:
  - idz.env is moved (renamed) to zee.env
- Since PTF UI60051:
  - ZUnit: A customizable member has been added:
    - JCL(AZUCSD)
- Since PTF UI58455:
  - Proclib: A customizable member has been added:
    - PROCLIB(ELAXF)
  - Proclib: Customizable members have been updated:
    - PROCLIB(ELAXF*): use ELAXF
    - PROCLIB(AZU*): use ELAXF
  - CARMA: A customizable member has been updated:
    - PARMLIB(CRACFG): add BYPASS-RESOURCE-LOCKING
- Since PTF UI56448:
  - CARMA: A customizable file has been updated:
    - CRASRV.properties: add system.exit
  - CARMA: New function:
    - Support for site-specific data in startup process
- Since PTF UI54674:
  - COBOL: Customizable files have been updated:
    - ELAXFCOC: add DD SYSOPTF
    - ELAXFCP1: add DD SYSOPTF

## Configurable files

Table 17 on page 83 shows an overview of z/OS Explorer Extensions files that are customized in version 14.2. The z/OS Explorer Extensions sample libraries, FEL.SFELSAMP and /usr/lpp/IBM/zee/samples/, contain more customizable members than listed here, such as sample CARMA source code and jobs to compile them.

Note: Sample job FELSETUP copies all listed members to different data sets and directories, default FEL.#CUST.* and /etc/zexpl/*.

*Table 17. Version 14.2 customizations*

| Member/File | Default location | Purpose | Migration notes |
|---|---|---|---|
| FELSETUP | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL to create data sets and directories, and populate them with customizable files | Updated to add actions for new files and paths |
| ELAXF | FEL.SFELSAMP<br>[FEL.#CUST.PROCLIB] | JCL INCLUDE to manage HLQs | New, customization is required |
| ELAXF* | FEL.SFELSAMP<br>[FEL.#CUST.PROCLIB] | JCL for remote project builds, and so on | Use ELAXF for HLQ management |
| FELRACF | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL for security definitions | None |
| FELCMPLY | FEL.SFELSAMP[<br>FEL.#CUST.JCL] | JCL for license compliance validation | None |
| FELSMF | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL for SMF reporting | None |
| CRA$VMSG | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL to create the CARMA message VSAM | None |
| CRA$VDEF | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL to create the CARMA configuration VSAM | None |
| CRA$VSTR | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL to create the CARMA custom information VSAM | None |
| CRA$VCAD | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL to create the CARMA configuration VSAM for CA Endevor® SCM RAM | None |
| CRA$VCAS | FEL.SFELSAMP<br>[FEL.#CUST.JCL] | JCL to create the CARMA custom information VSAM for CA Endevor® SCM RAM | None |
| CRASUBMT | FEL.SFELSAMP<br>[FEL.#CUST.CNTL] | CARMA batch startup CLIST | None |

| *Table 17. Version 14.2 customizations (continued)* | | | |
|---|---|---|---|
| **Member/File** | **Default location** | **Purpose** | **Migration notes** |
| CRASUBCA | FEL.SFELSAMP [FEL.#CUST.CNTL] | CARMA batch startup CLIST for CA Endevor® SCM RAM | None |
| CRACFG | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA interaction configuration for CA Endevor® SCM RAM | Add variable BYPASS-RESOURCE-LOCKING |
| CRABCFG | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA batch actions configuration for CA Endevor® SCM RAM | None |
| CRABATCA | FEL.SFELSAMP [FEL.#CUST.CNTL] | CARMA batch action JCL for CA Endevor® SCM RAM | None |
| CRASCL | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | Template SCL for CA Endevor® SCM | None |
| CRASHOW | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA configuration for CA Endevor® SCM RAM | None |
| CRATMAP | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA configuration for CA Endevor® SCM RAM | None |
| CRANDVRA | FEL.SFELPROC | CARMA allocation REXX for CA Endevor® SCM RAM | None |
| CRADYNDA | FEL.SFELPROC | CARMA allocation REXX for CA Endevor® SCM RAM | None |
| CRAALLOC | FEL.SFELPROC | CARMA allocation REXX | None |
| CRA#VSLM | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the SCLM RAM's message VSAM | None |
| CRA#ASLM | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the SCLM RAM's data sets | None |

| *Table 17. Version 14.2 customizations (continued)* | | | |
|---|---|---|---|
| **Member/File** | **Default location** | **Purpose** | **Migration notes** |
| CRA#VPDS | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the PDS RAM's message VSAM | None |
| CRA#UADD | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to merge RAM definitions | None |
| CRA#UQRY | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to extract RAM definitions | None |
| CRAXJCL | FEL.SFELSAMP [FEL.#CUST.ASM] | Sample source code for IRXJCL replacement | None |
| CRA#CIRX | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to compile CRAXJCL | None |
| FELTEP2 | FEL.SFELSAMP [FEL.#CUST.SQL] | SQL command input used by ELAXF* | None |
| FELTIAD | FEL.SFELSAMP [FEL.#CUST.SQL] | SQL command input used by ELAXF* | None |
| AZUALLOC | FEL.SEFLSAMP [FEL.#CUST.JCL] | JCL for ZUnit dataset allocation | None |
| AZUCSD | FEL.SEFLSAMP [FEL.#CUST.JCL] | JCL to define ZUnit to CICS | New, customization is optional |
| AZUZUDB2 | FEL.SFELSAMP [FEL.#CUST.PROCLIB] | JCL procedure for ZUnit | Use ELAXF for HLQ management |
| AZUZUNIT | FEL.SFELSAMP [FEL.#CUST.PROCLIB] | JCL procedure for ZUnit | Use ELAXF for HLQ management |
| FEKRNPLI | FEL.SFELSAMP [FEL.#CUST.CNTL] | REXX to call the PL/I compiler from within the preprocessor framework | None |
| IRZCSD | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to define Enterprise Service Tools to CICS | None |
| zee.env | /usr/lpp/IBM/zee/samples/ [/etc/zexpl/] | RSE environment variables | Renamed from idz.env, customization is required |
| CRASRV.properties | /usr/lpp/IBM/zee/samples/ [/etc/zexpl/] | CARMA configuration file | Add variable system.exit |

| Table 17. Version 14.2 customizations (continued) | | | |
|---|---|---|---|
| **Member/File** | **Default location** | **Purpose** | **Migration notes** |
| crastart.conf | /usr/lpp/IBM/zee/samples/ [/etc/zexpl/] | CARMA configuration file for CRASTART usage | None |
| crastart.endevor.conf | /usr/lpp/IBM/zee/samples/ [/etc/zexpl/] | CARMA configuration file for CRASTART usage for CA Endevor® SCM RAM | None |
| include.conf | /usr/lpp/IBM/zee/samples/ [/etc/zexpl/] | Forced includes for C/C++ content assist | None |

# IBM z/OS Source Code Analysis, FMID HAKGE20

- The FMID is renamed from IBM Developer for z Systems Host Utilities to IBM z/OS Source Code Analysis
- The default SMP/E install location for MVS did not change and remains AKG.*.
- The default SMP/E install location for z/OS UNIX components changed from /usr/lpp/IBM/idzutil/* to /usr/lpp/IBM/akg/*.
- The customizable path /var/idzutil has been updated (renamed) to /var/akg.

## Configurable files

Table 18 on page 86 shows an overview of files that are customized in version 14.2. The IBM z/OS Source Code Analysis sample libraries, AKG.SAKGSAMP and /usr/lpp/IBM/akg/samples, contain more customizable members than listed here, such as sample code review post-processing script.

**Note:** Sample job AKGSETUP copies all listed members to different data sets, default AKG.#CUST.*.

| Table 18. Version 14.2 customizations | | | |
|---|---|---|---|
| **Member or File** | **Default location** | **Purpose** | **Migration notes** |
| AKGSETUP | AKG.SAKGSAMP [AKG.#CUST.JCL] | JCL to create data sets, and populate them with customizable files | Update default path names |
| AKGCC | AKG.SAKGSAMP [AKG.#CUST.PROCLIB] | JCL for code coverage | Update default path names |
| AKGCR | AKG.SAKGSAMP [AKG.#CUST.PROCLIB] | JCL for code review | Update default path names |
| AKGCRADD | AKG.SAKGSAMP [AKG.#CUST.JCL] | JCL to add third-party code to code review | Update default path names |

# Migrate from version 14.0 to version 14.1

These notes are for a migration from a base version 14.0 to version 14.1. It includes changes that are already documented as part of version 14.0 maintenance. The changes that are part of the maintenance stream, and thus possibly already implemented, are marked with the release where they were introduced.

## IBM Developer for z Systems, FMID HHOPE10

- The default SMP/E install location for MVS and z/OS UNIX components did not change and remain `FEL.*` and `/user/lpp/IBM/idz/*`.
- Customization: The FELSETUP JCL now processes the new members:
  - JCL(AZUALLOC)
  - JCL(FELCMPLY)
  - CNTL(FELEDTMC)
- New customizable members have been added:
  - FELEDTMC: execute ISPF EDIT macros
- Since version 14.0.0.5
  - CARMA: Customizable files have been updated:
    - CRASRV.properties: add variable connect.timeout
- Since version 14.0.0.4
  - Reporting: Customizable members have been added:
    - JCL(FELCMPLY): verify VU-license compliance
  - Reporting: Customizable members have been updated:
    - JCL(FELSMF)
  - CARMA: Customizable files have been updated:
    - CRADYNDA: add RECFM support
- Since version 14.0.0.1
  - ZUnit: Customizable members have been updated:
    - AZUALLOC: COBOL listing data set now allocated as FBA133

### Configurable files

Table 19 on page 87 shows an overview of IBM Developer for z Systems files that are customized in version 14.1. The Developer for z Systems sample libraries, `FEL.SFELSAMP`, `FEL.SFELSAMV` and `/usr/lpp/IBM/idz/samples/`, contain more customizable members than listed here, such as sample CARMA source code and jobs to compile them.

Note: Sample job FELSETUP copies all listed members to different data sets and directories, default `FEL.#CUST.*` and `/etc/zexpl/*`.

*Table 19. Version 14.1 customizations*

| Member/File | Default location | Purpose | Migration notes |
|---|---|---|---|
| FELSETUP | `FEL.SFELSAMP`<br>`[FEL.#CUST.JCL]` | JCL to create data sets and directories, and populate them with customizable files | Updated to add actoins for new files |

| *Table 19. Version 14.1 customizations (continued)* | | | |
|---|---|---|---|
| **Member/File** | **Default location** | **Purpose** | **Migration notes** |
| ELAXF* | FEL.SFELSAMP [FEL.#CUST.PROCLIB] | JCL for remote project builds, and so on | None |
| FELRACF | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL for security definitions | None |
| FELCMPLY | FEL.SFELSAMP[ FEL.#CUST.JCL] | JCL for license compliance validation | New, customization is optional |
| FELSMF | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL for SMF reporting | New, customization is optional |
| CRA$VMSG | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the CARMA message VSAM | None |
| CRA$VDEF | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the CARMA configuration VSAM | None |
| CRA$VSTR | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the CARMA custom information VSAM | None |
| CRA$VCAD | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the CARMA configuration VSAM for CA Endevor® SCM RAM | None |
| CRA$VCAS | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the CARMA custom information VSAM for CA Endevor® SCM RAM | None |
| CRASUBMT | FEL.SFELSAMP [FEL.#CUST.CNTL] | CARMA batch startup CLIST | None |
| CRASUBCA | FEL.SFELSAMP [FEL.#CUST.CNTL] | CARMA batch startup CLIST for CA Endevor® SCM RAM | None |
| CRACFG | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA interaction configuration for CA Endevor® SCM RAM | None |

*Table 19. Version 14.1 customizations (continued)*

| Member/File | Default location | Purpose | Migration notes |
|---|---|---|---|
| CRABCFG | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA batch actions configuration for CA Endevor® SCM RAM | None |
| CRABATCA | FEL.SFELSAMP [FEL.#CUST.CNTL] | CARMA batch action JCL for CA Endevor® SCM RAM | None |
| CRASCL | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | Template SCL for CA Endevor® SCM | None |
| CRASHOW | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA configuration for CA Endevor® SCM RAM | None |
| CRATMAP | FEL.SFELSAMP [FEL.#CUST.PARMLIB] | CARMA configuration for CA Endevor® SCM RAM | None |
| CRANDVRA | FEL.SFELPROC | CARMA allocation REXX for CA Endevor® SCM RAM | None |
| CRADYNDA | FEL.SFELPROC | CARMA allocation REXX for CA Endevor® SCM RAM | Add RECFM support |
| CRAALLOC | FEL.SFELPROC | CARMA allocation REXX | None |
| CRA#VSLM | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the SCLM RAM's message VSAM | None |
| CRA#ASLM | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the SCLM RAM's data sets | None |
| CRA#VPDS | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to create the PDS RAM's message VSAM | None |
| CRA#UADD | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to merge RAM definitions | None |
| CRA#UQRY | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to extract RAM definitions | None |

| *Table 19. Version 14.1 customizations (continued)* | | | |
|---|---|---|---|
| **Member/File** | **Default location** | **Purpose** | **Migration notes** |
| CRAXJCL | FEL.SFELSAMP [FEL.#CUST.ASM] | Sample source code for IRXJCL replacement | None |
| CRA#CIRX | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to compile CRAXJCL | None |
| FELTEP2 | FEL.SFELSAMP [FEL.#CUST.SQL] | SQL command input used by ELAXF* | None |
| FELTIAD | FEL.SFELSAMP [FEL.#CUST.SQL] | SQL command input used by ELAXF* | None |
| AZUALLOC | FEL.SEFLSAMP [FEL.#CUST.JCL] | JCL for ZUnit dataset allocation | COBOL LISTING definition updated |
| AZUZUDB2 | FEL.SFELSAMP [FEL.#CUST.PROCLIB] | JCL procedure for ZUnit | None |
| AZUZUNIT | FEL.SFELSAMP [FEL.#CUST.PROCLIB] | JCL procedure for ZUnit | None |
| FEKRNPLI | FEL.SFELSAMP [FEL.#CUST.CNTL] | REXX to call the PL/I compiler from within the preprocessor framework | None |
| IRZCSD | FEL.SFELSAMP [FEL.#CUST.JCL] | JCL to define Enterprise Service Tools to CICS | None |
| idz.env | /usr/lpp/IBM/idz/samples/ [/etc/zexpl/] | RSE environment variables | None |
| CRASRV.properties | /usr/lpp/IBM/idz/samples/ [/etc/zexpl/] | CARMA configuration file | Add variable connect.timeout |
| crastart.conf | /usr/lpp/IBM/idz/samples/ [/etc/zexpl/] | CARMA configuration file for CRASTART usage | None |
| crastart.endevor.conf | /usr/lpp/IBM/idz/samples/ [/etc/zexpl/] | CARMA configuration file for CRASTART usage for CA Endevor® SCM RAM | None |
| include.conf | /usr/lpp/IBM/idz/samples/ [/etc/zexpl/] | Forced includes for C/C++ content assist | None |

## IBM Developer for z Systems Host Utilities, FMID HAKGE10

- The default SMP/E install location for MVS and z/OS components did not change and remain `AKG.*` and `/usr/lpp/IBM/idzutil/*`.
- Code Review: Customizable files have been updated:
  - PROCLIB(AKGCR): Java version updated to Java v8.0

### Configurable files

Table 20 on page 91 shows an overview of files that are customized in version 14.1. The Developer for z Systems Host Utilities sample libraries, `AKG.SAKGSAMP` and `/usr/lpp/IBM/idzutil/samples`, contain more customizable members than listed here, such as sample code review post-processing script.

**Note:** Sample job AKGSETUP copies all listed members to different data sets, default `AKG.#CUST.*`.

*Table 20. Version 14.1 customizations*

| Member or File | Default location | Purpose | Migration notes |
|---|---|---|---|
| AKGSETUP | `AKG.SAKGSAMP [AKG.#CUST.JCL]` | JCL to create data sets, and populate them with customizable files | None |
| AKGCC | `AKG.SAKGSAMP [AKG.#CUST.PROCLIB]` | JCL for code coverage | None |
| AKGCR | `AKG.SAKGSAMP [AKG.#CUST.PROCLIB]` | JCL for code review | Update Java version to J8.0 |
| AKGCRADD | `AKG.SAKGSAMP [AKG.#CUST.JCL]` | JCL to add third-party code to code review | None |

# Notices

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*

© Copyright IBM Corp. 2000, 2024

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-IBMproducts. 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.

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.

## Programming interface information

## Trademarks

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

## Terms and conditions for product documentation

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.

SC27-9933-03