1.2

IBM Z Monitoring Configuration Manager

SC28-3147

Note:

Before using this information and the product it supports, read the "Notices" topic at the end of this information.

Last updated: 2024-04-19

This edition applies to Version 1 Release 2 of IBM Z Monitoring Configuration Manager and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this information

IBM Z® Monitoring Configuration Manager (also referred to as Monitoring Configuration Manager) is a tool that configures an OMEGAMON runtime environment from a set of parameters that you specify. This process is easier and faster than using the legacy PARMGEN, with its many parameters, for configuration.

These topics provide instructions for using Monitoring Configuration Manager to perform the following tasks.

- Develop batch jobs to run the Create, Discover, Migrate, Generate, and Delete actions. See “Batch interface” on page 31 for details.
- Select which parameters you want to use with Monitoring Configuration Manager. See“Parameters” on page 69 for a description of valid parameters.
- Specify a library to contain the runtime environment definitions (RTEDEF). See “Runtime environment definition (RTEDEF) library” on page 107 for more information.
- View troubleshooting material and error messages you might see when using Monitoring Configuration Manager. See “Troubleshooting” on page 147 and “Messages” on page 151 for more information.

Products supported by Configuration Manager

These products can be configured using Monitoring Configuration Manager, regardless of the product being part of a suite or pack offering or purchased as a standalone point product.

Each product is listed with its corresponding product code.

- Tivoli Enterprise Monitoring Server 6.3 (DS)
- OMEGAMON Enhanced 3270 User Interface 7.5 (OB)
- IBM OMEGAMON Dashboard Edition on z/OS 5.5 (WO)
- IBM Z OMEGAMON Integration Monitor 5.6 (WO)
- IBM OMEGAMON for CICS on z/OS 5.5 (C5)
- IBM Z OMEGAMON for CICS 5.6 (CICS TS: C5; CICS TG: GW)
- IBM Tivoli OMEGAMON XE for Db2 Performance Expert on z/OS 5.4 (D5)
- IBM Tivoli OMEGAMON XE for Db2 Performance Monitor on z/OS 5.4 (D5)
- IBM OMEGAMON for Db2 Performance Expert on z/OS 5.5 (D5)
- IBM OMEGAMON for IMS on z/OS 5.5 (I5)
- IBM Z OMEGAMON for JVM 5.5 (JJ)
- IBM Z OMEGAMON Runtime Edition for JVM 5.5 (JJ)
- IBM Z OMEGAMON AI for JVM, 6.1 (JJ)
- IBM OMEGAMON for Messaging on z/OS 7.5 (MQ: MQ; Integration Bus: QI)
- IBM OMEGAMON for Networks on z/OS 5.5 (N3)
- IBM Z OMEGAMON Network Monitor 5.6 (N3)
- IBM Z OMEGAMON AI for Networks 6.1 (N3)
- IBM OMEGAMON for Storage on z/OS 5.5 (S3)
- IBM OMEGAMON for z/OS 5.5 (M5)
- IBM Z OMEGAMON Monitor for z/OS 5.6 (M5)
- IBM Z OMEGAMON AI for z/OS 6.1 (M5)
- IBM Z NetView Enterprise Management Agent 6.3 (NA)
- IBM Z NetView Enterprise Management Agent 6.4 (NA)
- IBM Tivoli Advanced Allocation Management for z/OS 3.3 (RJ)
- IBM Tivoli Advanced Audit for DFSMShsm 2.6 (RG)
- IBM Tivoli Advanced Backup and Recovery for z/OS 2.4 (RV)
- IBM Tivoli Advanced Catalog Management for z/OS 2.6 (RN)
- IBM Tivoli Advanced Reporting and Management for DFSMShsm 2.6 (RH)
- IBM Tivoli Automated Tape Allocation Manager for z/OS 3.3 (RK)
- IBM Tivoli Composite Application Manager (ITCAM) for Application Diagnostics Agent 7.1.0 (YN)
- IBM Tivoli Composite Application Manager (ITCAM) for Application Diagnostics Agent 7.1.1 (YN)

For links to the documentation for these products, see Where to find information.

Introduction to Configuration Manager

IBM Z Monitoring Configuration Manager, also known as Monitoring Configuration Manager or Configuration Manager, is a tool that configures an OMEGAMON runtime environment from a set of parameters that you specify.

A runtime environment (RTE) consists of the started tasks and related members, including MVS™ data sets and z/OS® UNIX System Services files, that are required to monitor subsystems on a z/OS LPAR. These started tasks and related members are collectively known as *runtime members*.

To generate the runtime members for a runtime environment, you configure a set of parameters, and then you run a single Monitoring Configuration Manager job. Parameters are name-value pairs stored as plain text. The following figure illustrates this basic concept:

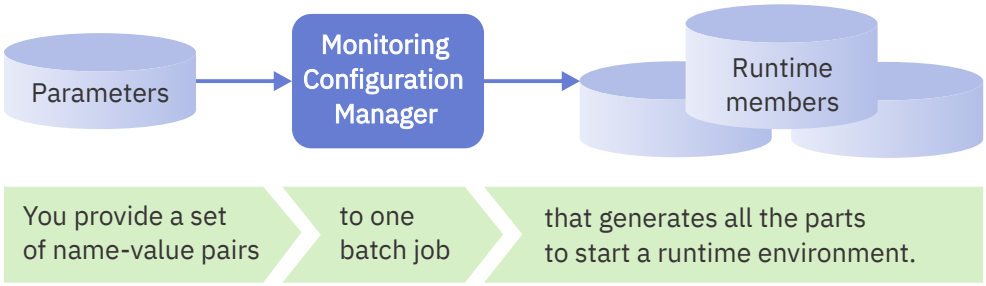

Figure 1. The basic concept: parameters in, one job, runtime members out

Generating runtime members from parameters is the main purpose of Monitoring Configuration Manager. This is known as the **GENERATE** action.

Monitoring Configuration Manager can also perform other actions:

- The **CREATE** action creates an initial set of parameters for a runtime environment.
- The **DISCOVER** action discovers subsystems on an LPAR, and then creates corresponding parameters to configure a runtime environment to monitor those subsystems.
- The **DELETE** action deletes the runtime members for a runtime environment.

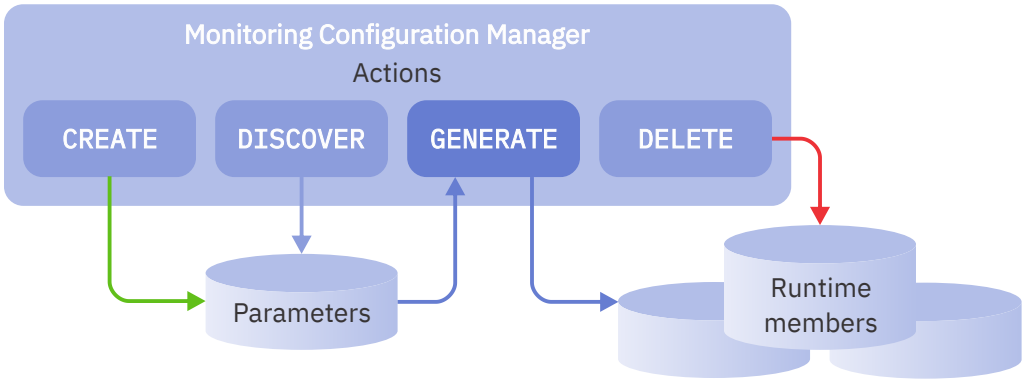

Figure 2. Actions

In addition to the actions shown in the diagram:

- The **MIGRATE** action imports existing PARMGEN RTE configuration settings from a specific `WCONFIG` member.
- The **PACKAGE** and **DEPLOY** actions support a convenient rollout from a single LPAR to other runtime environments.

Related tasks

Creating your first, minimal runtime environment

If you are a first-time user of IBM Z Monitoring Configuration Manager, creating a minimal runtime environment is a good place to start. This example consists of a z/OS agent, a monitoring server, and an enhanced 3270 user interface. You can logon to the enhanced 3270 user interface to view data from the z/OS agent.

Related reference

Runtime environment definition (RTEDEF) library

A runtime environment definition is a set of parameters. Parameters are stored in a runtime environment definition (RTEDEF) library. The set of parameters for each runtime environment is organized into several RTEDEF members.

Comparison with PARMGEN

IBM Z Monitoring Configuration Manager evolved from PARMGEN. If you understand PARMGEN, then a comparison can help you to quickly understand Monitoring Configuration Manager.

You can use either Monitoring Configuration Manager or PARMGEN to configure runtime environments for the supported OMEGAMON agents. Monitoring Configuration Manager and PARMGEN use mainly the same parameters. However, Monitoring Configuration Manager radically simplifies the process of configuring runtime environments from those parameters.

You must decide whether to use Monitoring Configuration Manager or PARMGEN

You can use Monitoring Configuration Manager to generate runtime members for some runtime environments, PARMGEN for others, and run them in the same monitoring topology, communicating with the same hub monitoring server. In that sense, you can use Monitoring Configuration Manager and PARMGEN alongside each other.

However, you cannot use Monitoring Configuration Manager and PARMGEN interchangeably to generate runtime members for a runtime environment from the same set of parameters. For each runtime environment, you must decide whether to use Monitoring Configuration Manager or PARMGEN.

Note: You can move PARMGEN data to Configuration Manager using the **MIGRATE** action.

Overview

The following table provides an overview of the differences between Monitoring Configuration Manager and PARMGEN.

Table 1. Monitoring Configuration Manager versus PARMGEN: overview

Monitoring Configuration Manager	PARMGEN
Batch-only interface.	Combination of ISPF user interface and batch. You navigate ISPF panels to select which job to submit.
The same simple, concise JCL for all actions.	Different jobs for different actions. Complex, long JCL. JCL members tailored for each runtime environment must be created, stored, and potentially recreated, depending on the situation.
To generate runtime members from parameters, you submit ***one* job**.	To generate runtime members from parameters, you use ISPF panels to submit **a series of jobs**. Some jobs submit other jobs. You need to check the output from each job, and then return to the ISPF panels to submit the next job in the series.
To generate runtime members, you submit the same job in all situations. Enhancements introduced by Monitoring Configuration Manager, including performance improvements and the streamlining of previously separate stages into a single job, removes the need for users to decide which stages to run in different situations.	You need to understand which job, or series of jobs, to submit in different situations. For example, you need to understand which jobs to run to update a runtime environment after applying SMP/E maintenance to your OMEGAMON agents.

Table 1. Monitoring Configuration Manager versus PARMGEN: overview (continued)

Monitoring Configuration Manager	PARMGEN
You only need to know about the input parameters and the output runtime members. Monitoring Configuration Manager insulates you from the underlying complexity.	In addition to understanding the inputs and outputs, you also need to understand the details of the process that generates runtime members. For example, you need to understand the difference between interim staging (IK*) libraries, work (WK*) libraries, and the final output runtime (RK*) libraries. You also need to understand which stages of the process, and which jobs, affect each of those libraries.
Sparse configuration profiles containing only the parameters you need. The initial configuration profile members contain only **a few dozen parameters**. This is all you need to use basic functions if you are content with default parameter values.	Comprehensive configuration profiles containing all parameters for all agents. You edit a configuration profile member containing **hundreds of parameters** interspersed with multiline comments.
Integrated subsystem discovery.	Requires IBM Discovery Library Adapter for z/OS (DLA).
Available for all agents of the IBM Z Monitoring Suite and the IBM Z NetView Enterprise Management Agent, as well as for IBM OMEGAMON for z/OS version 5.5.0 or later, IBM OMEGAMON for Networks on z/OS V5.5.0 or later, and IBM Z OMEGAMON Integration Monitor V5.5.0 or later. In addition, support is provided for all agents that are part of the IBM Z Service Management Suite. This includes the same list of agents as above, except for Integration Monitor. Point product installations are also supported for the aforementioned agents/products.	Available with IBM Z Monitoring Suite and other products.

Details

The following table describes some differences in the implementation details between Monitoring Configuration Manager and PARMGEN.

For a comprehensive list of parameters that have different default values in Monitoring Configuration Manager and PARMGEN, see “Parameters with different default values than PARMGEN” on page 80.

Text in *italics* represents a parameter value. For example, *rte_plib_hilev* represents the value of the **RTE_PLIB_HILEV** parameter.

Table 2. Monitoring Configuration Manager versus PARMGEN: details

Monitoring Configuration Manager	PARMGEN
Stores parameters and variables in: *rte_plib_hilev*.RTEDEF Each RTEDEF library can contain definitions for multiple runtime environments, customized to run on multiple LPARs. **Note:** The Configuration Manager **GENERATE** action creates an *rte_plib_hilev*.*rte_name*.WCONFIG(*rte_name*) member that is similar to the member created by PARMGEN, with one key difference: the member created by Configuration Manager contains default parameter values; it does not reflect the values in your RTEDEF library members. The WCONFIG data sets created by Configuration Manager, as a whole, should be considered a *black box*.	Stores parameters in: *rte_plib_hilev*.*rte_name*.WCONFIG Each WCONFIG library contains the definition for a single runtime environment, with limited flexibility to customize that definition to run on multiple LPARs. Stores variables in: *gbl_user_jcl*
Organizes parameters into members according to their prefix and whether they apply to all LPARs or only to a specific LPAR. Provides a well-defined member naming convention so you know which parameters to store where and which parameters take precedence. Similarly, Monitoring Configuration Manager organizes variables into members that enable different values for each LPAR. You can configure a runtime environment using a combination of parameters that are common to all LPARs and parameters that apply only to a specific LPAR.	Mixes runtime environment (**RTE_***) parameters and product-specific (**Kpp_***) parameters in a single large member, WCONFIG(*rte_name*). You can divide this monolithic member into smaller members, but you need to manage this yourself for each runtime environment. You need to define your own member naming convention, and then edit the WCONFIG($SYSIN) member to include the members in PARMGEN processing and define their order of precedence. No built-in support for LPAR-specific parameter values beyond using variables, such as SYSNAME, that have LPAR-specific values.
Enables you to define LPAR-specific parameter values without using variables.	Uses variables to customize configuration profiles for different LPARs.
Uses the parameters in the first row of a table as the defaults for subsequent rows. If you omit a parameter from a subsequent row, that row uses the value from the first row. This enables you to define more concise "sparse" parameter tables with less duplication of values.	You must specify parameter values for each row in a table of parameters.
By default, writes system library members to the same high-level qualifiers as other non-VSAM runtime members: *rte_hilev*.SYS1.* where * is the system library low-level qualifier: PROCLIB, VTAMLIB, or VTAMLST	By default, writes system library members directly to: SYS1.*

Table 2. Monitoring Configuration Manager versus PARMGEN: details (continued)

Monitoring Configuration Manager	PARMGEN
Writes concise started tasks with minimal comments. **Tip:** Monitoring Configuration Manager writes concise started tasks to: *rte_hilev*.SYS1.PROCLIB and versions with verbose comments to the same location used by PARMGEN: *rte_plib_hilev*.*rte_name*.RKANSAMU *rte_plib_hilev*.*rte_name*.RKD2SAM (for Db2®)	Writes started tasks with verbose comments.
Sets the default value of the **RTE_USS_DIR** parameter to /var/rtehome.	Sets the default value of the **RTE_USS_DIR** parameter to /rtehome, a subdirectory of the root directory. Creating a new subdirectory of the root directory is bad practice.
In the z/OS UNIX System Services file system, by default writes only to *rte_uss_dir*/*rte_name*	By default, writes to various z/OS UNIX directories.
Sets the default value of the **RTE_TYPE** parameter to SHARING and **RTE_SHARE** to SMP. By default, runtime environments refer to some runtime members, such as load modules, in the SMP/E installation target library, rather than creating a full copy of those members in the runtime environment's own runtime libraries.	Sets the default value of the **RTE_TYPE** parameter to FULL. By default, the runtime environment runtime members include a full copy of all members required from the SMP/E installation target library.
Provides the ability to create one or more copies of SMP/E target libraries from which you can create or update your runtime environments. Sharing with an SMP/E target in reality is sharing with an SMP/E target copy. The Configuration Manager *target copy* feature copies only the data sets needed for products that are selected for configuration into the target copy libraries. For **RTE_SHARE**, you can only specify SMP for a sharing runtime environment. Configuration Manager does not use base runtime environments (as in PARMGEN).	Provides the ability to use a set of static base libraries in a base runtime environment from which a sharing-with-base runtime environment obtains read-only runtime libraries. For **RTE_SHARE**, you can specify the name of base or full runtime environment from where a sharing runtime environment obtains its base library information, or you can specify SMP to share SMP/E target libraries.

Preparing to use Configuration Manager

Review prerequisites and other considerations before you start using Configuration Manager.

This section provides information about the following items, which are prerequisites or should be considered before using Configuration Manager:

- You need to know where your SMP/E installation target library is located. See Location of SMP/E installation target libraries.
- The TSO user ID that will be used to run Configuration Manager jobs needs some special access privileges. See Access privileges for TSO user ID.
- Consider the naming convention for your runtime environments. See Naming your runtime environment.
- Review other known issues. See Issues to be aware of before using Configuration Manager.

Location of SMP/E installation target libraries

You need to know where your target libraries are installed:

- On MVS: The high-level qualifiers of the SMP/E target libraries, such as TKANMOD.
- On z/OS UNIX System Services: For products that require it, such as OMEGAMON for CICS and OMEGAMON for JVM, the path of the SMP/E target directory that is defined in the SMP/E installation jobs by ddname TKANJAR. The default directory path is `/usr/lpp/kan/bin/IBM`.

Typical best practice is to make a copy of the original SMP/E-managed locations and refer to the copies. This enables you to manage when to introduce changes in the original SMP/E-managed locations into your environment.

Access privileges for TSO user ID

The TSO user ID that you plan to run Configuration Manager jobs (for example, your own user ID) must have the following access privileges:

- Read access to the target libraries and the z/OS UNIX directory defined by the TKANJAR ddname.
- Read access to the following z/OS System Authorization Facility (SAF) resources in the FACILITY class:

```
BPX.FILEATTR.APF
BPX.FILEATTR.PROGCTL
```

You do not need z/OS UNIX superuser privileges to run Configuration Manager.

Naming your runtime environment

When you define your runtime environment, it is recommended to use a naming convention that will not interfere with your system libraries. Consider meaningful names that will easily distinguish and isolate your OMEGAMON runtime environments on your system.

Configuration Manager uses the values of **RTE_NAME** and **RTE_PLIB_HILEV** to set the name of the runtime environment and runtime environment definition library. It is recommended that the combined length of these parameters does not exceed 28 characters. For more information, see “Creating your first, minimal runtime environment” on page 13.

Issues to be aware of before using Configuration Manager

When using Configuration Manager, especially the **GENERATE** action, make sure the job does not use a Batch Optimization tool. These tools are known to exhaust below-the-line storage and fail the job with ABEND878-10.

Defining the OMEGAMON subsystem to z/OS

Some OMEGAMON® monitoring agents depend on the OMEGAMON subsystem. Before starting a runtime environment that contains any of these monitoring agents, you must define the OMEGAMON subsystem to z/OS.

Before you begin

Check whether the OMEGAMON subsystem has already been defined to z/OS.

For example, issue the following **DISPLAY** MVS system command to list subsystems:

```
D SSI
```

The default name of the OMEGAMON subsystem is CNDL.

About this task

IBM Z Monitoring Configuration Manager does not depend on the OMEGAMON subsystem. Defining the OMEGAMON subsystem is not a prerequisite for using Monitoring Configuration Manager.

However, the OMEGAMON subsystem *is* a prerequisite for starting some of the runtime environments that you create with Monitoring Configuration Manager.

The following OMEGAMON monitoring agents depend on the OMEGAMON subsystem:

- CICS® (optional)
- Db2
- IMS
- Storage
- z/OS

You must define the OMEGAMON subsystem on each LPAR where you plan to start runtime environments that contain any of these monitoring agents.

When you create a runtime environment that contains one of these monitoring agents, the generated runtime members include a started task that starts the OMEGAMON subsystem.

Optionally, you can configure your LPAR to start the OMEGAMON subsystem whenever z/OS restarts (IPL).

Procedure

1. Copy the OMEGAMON subsystem initialization module KCNDLINT from the target library TKANMOD to a library in the linklist.

   For example, SYS1.LINKLIB.
2. Refresh the library lookaside (LLA) library directory indexes.

   Issue the following **MODIFY** MVS system command:

   ```
   F LLA,REFRESH
   ```
3. Dynamically define the OMEGAMON subsystem to z/OS.

   Issue the following **SETSSI** MVS system command:

   ```
   SETSSI ADD,SUBNAME=CNDL,INITRTN=KCNDLINT,INITPARM='SSPROC=OMEGCN'
   ```

   where:

   - CNDL is the subsystem name.

CNDL is the default value of the runtime environment parameter **RTE_KCNSTR00_SSID**.

The values of **SUBNAME** and **RTE_KCNSTR00_SSID** must match.

- OMEGCN, the name of the procedure that initializes the subsystem, is the value of the runtime environment parameter **RTE_CANSCN_STC**.

  The default value of **RTE_CANSCN_STC** consists of the value of the **RTE_STC_PREFIX** parameter followed by the suffix CN. The value of **RTE_STC_PREFIX** in the initial set of parameters created by Monitoring Configuration Manager is OMEG (an abbreviation of OMEGAMON).

  The **SSPROC** parameter of the **SETSSI** command and the runtime environment parameter **RTE_CANSCN_STC** must match.

If you do not want the subsystem address space to be started immediately, omit **INITPARM**.

4. Define the OMEGAMON subsystem in the IEFSSN*xx* member of the SYS1.PARMLIB library.

   For example:

   ```
   SUBSYS SUBNAME(CNDL) INITRTN(KCNDLINT) INITPARM('SSPROC=OMEGCN')
   ```

   The **SETSSI** command in the previous step dynamically defines the subsystem so that it is available immediately. Defining the subsystem in the IEFSSN*xx* member defines the subsystem during z/OS initialization (IPL), so that you do not have to reissue the **SETSSI** command after each IPL.

   If you do not want the subsystem address space to be started at IPL, omit **INITPARM**.

Creating your first, minimal runtime environment

If you are a first-time user of IBM Z Monitoring Configuration Manager, creating a minimal runtime environment is a good place to start. This example consists of a z/OS agent, a monitoring server, and an enhanced 3270 user interface. You can logon to the enhanced 3270 user interface to view data from the z/OS agent.

Before you begin

Read the prerequisites for using Monitoring Configuration Manager.

The OMEGAMON subsystem must be defined to z/OS.

About this task

Here is a diagram of the minimal runtime environment created by the following procedure:

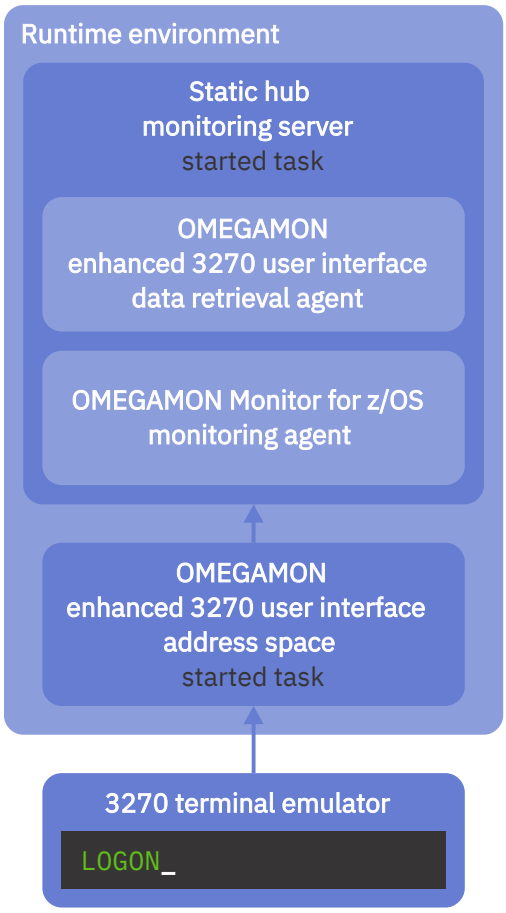

Figure 3. Overview of a minimal runtime environment

To create this minimal runtime environment, you follow the same procedure you would follow to create any runtime environment. The difference is that a minimal runtime environment involves setting fewer parameters, and involves fewer tasks to complete the configuration after running Monitoring Configuration Manager.

Procedure

1. Submit a job that performs the **CREATE** action of Monitoring Configuration Manager.

The **CREATE** action creates a runtime environment definition library, *rte_plib_hilev*.RTEDEF, and populates it with an initial set of parameters.

Example JCL:

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            CREATE
RTE_NAME          <rte_name>
RTE_PLIB_HILEV    <rte_plib_hilev>
/*
```

*Figure 4. Example JCL to perform the **CREATE** action*

Similar JCL is supplied in the KFJJMCM member of the TKANSAM target library.

Note: Refer to this sample member, KFJJMCM, for any updates to the parameters. Any new or changed parameters will be listed in this member and can be customized according to the action that you want to run.

Edit the example job statement to match your site's standards. For example, for job name, class, and message class. Consider changing the example job name prefix, UID, to your TSO user ID.

Replace the placeholders in the example JCL with appropriate values:

<lpar>
Run Monitoring Configuration Manager actions on the LPAR where you will start the runtime environment.

For example, if your site uses JES2, insert a **SYSAFF** job parameter after the **JOB** statement to ensure that the job runs on that LPAR.

<tlib_hlq>
The high-level qualifiers of the target libraries.

<rte_name>
Runtime environment name, 1 - 8 characters.

Monitoring Configuration Manager uses this name for various purposes, including:

- MVS member names
- MVS data set name qualifiers
- z/OS UNIX System Services directory name, all uppercase

<rte_plib_hilev>
The high-level qualifiers of the runtime environment definition library:

rte_plib_hilev.RTEDEF

Monitoring Configuration Manager uses the values of **RTE_NAME** and **RTE_PLIB_HILEV** to set the default value of other parameters, such as **RTE_HILEV** and **RTE_VSAM_HILEV**, that are used for data set names. To avoid exceeding the z/OS 44-character limit for data set names, the combined length of **RTE_NAME** and **RTE_PLIB_HILEV** should not exceed 28 characters. For example, if **RTE_NAME** is 8 characters, then **RTE_PLIB_HILEV** should not exceed 20 characters.

Tip: After running a Monitoring Configuration Manager job, check the KCIPRINT sysout data set.

2. Edit the parameters in the *rte_plib_hilev*.RTEDEF library to configure a static hub monitoring server, a z/OS agent, and an enhanced 3270 user interface.

In the RTEDEF(*rte_name*) member, set the following **CONFIGURE_*** parameters to Y:

CONFIGURE_TEMS_KDS
Configures a monitoring server (Tivoli® Enterprise Monitoring Server, or TEMS)

CONFIGURE_ZOS_KM5
Configures a z/OS monitoring agent

CONFIGURE_E3270UI_KOB
Configures an enhanced 3270 user interface address space

Either delete all other **CONFIGURE_*** parameters or set them to N.

The following diagram shows how these **CONFIGURE_*** parameters, and the default values of related parameters, configure the topology of the runtime environment:

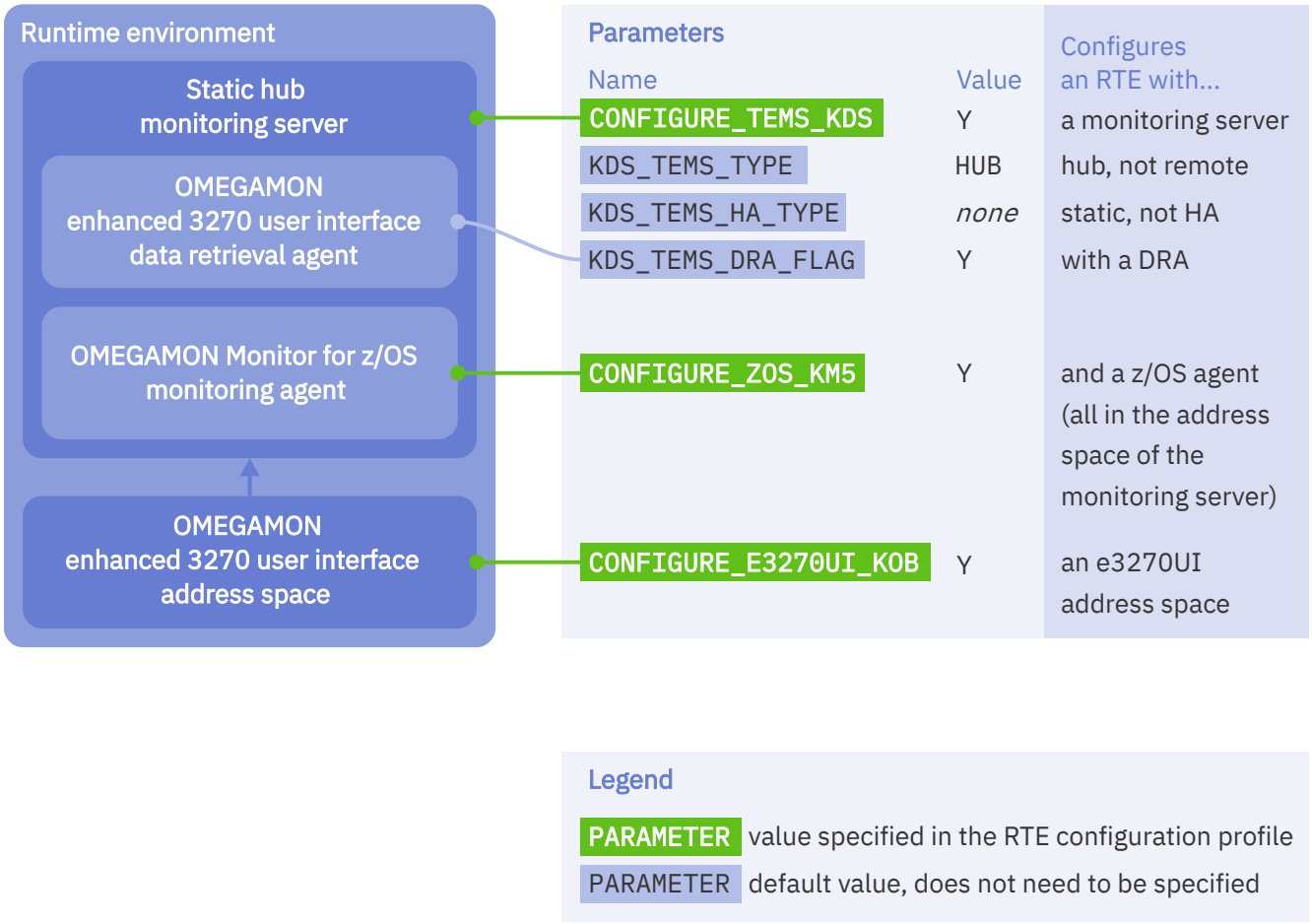

*Figure 5. **CONFIGURE_*** parameters for a minimal runtime environment*

Also in the RTEDEF(*rte_name*) member, set the following parameters to match your site-specific standards:

RTE_STC_PREFIX
1- to 4-character prefix of the started task names for this runtime environment. The value in the initial set of parameters is OMEG.

RTE_VTAM_APPLID_PREFIX
Prefix of the VTAM® applids in this runtime environment. For this minimal runtime environment, there is only one VTAM application: the enhanced 3270 user interface.

Each VTAM application in a runtime environment has a corresponding parameter for the VTAM applid. The default values of these parameters are the value of **RTE_VTAM_APPLID_PREFIX** followed by an application-specific suffix.

In the initial set of parameters created by the **CREATE** action, the value of **RTE_VTAM_APPLID_PREFIX** is OM*xx*, where *xx* is the value of the z/OS static system symbol **SYSCLONE**. **SYSCLONE** is a 1- or 2-character shorthand notation for the system (LPAR) name. This

value is one example of why you need to run Monitoring Configuration Manager actions on the LPAR where you will start the runtime environment.

If you use these values, then the default VTAM applid for the enhanced 3270 user interface is 0M*xx*0BAP. For example, if the system (LPAR) name is Z0S1, then the VTAM applid is 0MS10BAP.

RTE_USS_RTEDIR
The path of the z/OS UNIX directory where you want Monitoring Configuration Manager to write runtime files required by the started tasks.

The TSO user ID that runs Monitoring Configuration Manager jobs must have permission to write to this directory, otherwise the **GENERATE** action will fail.

RTE_TCP_PORT_NUM
The TCP/IP port number on which the monitoring server will listen.

Tip: Later steps in this procedure describe how to activate VTAM resources and APF-authorize libraries. If you insert the parameter **RTE_X_STC_INAPF_INCLUDE_FLAG** Y in the RTEDEF(*rte_name*) member, then the started tasks include a member that performs these steps for you.

In the RTEDEF(GBL$PARM) member, set the following parameter:

GBL_HFS_JAVA_DIR1
The z/OS UNIX path of the Java™ home directory.

The following diagram shows parameters that configure identifiers and values used by the runtime environment. Notice how the **RTE** parameters determine the default values of other parameters. For example, the default value of the **KDS_TEMS_STC** parameter is the value of **RTE_STC_PREFIX** followed by the suffix DS.

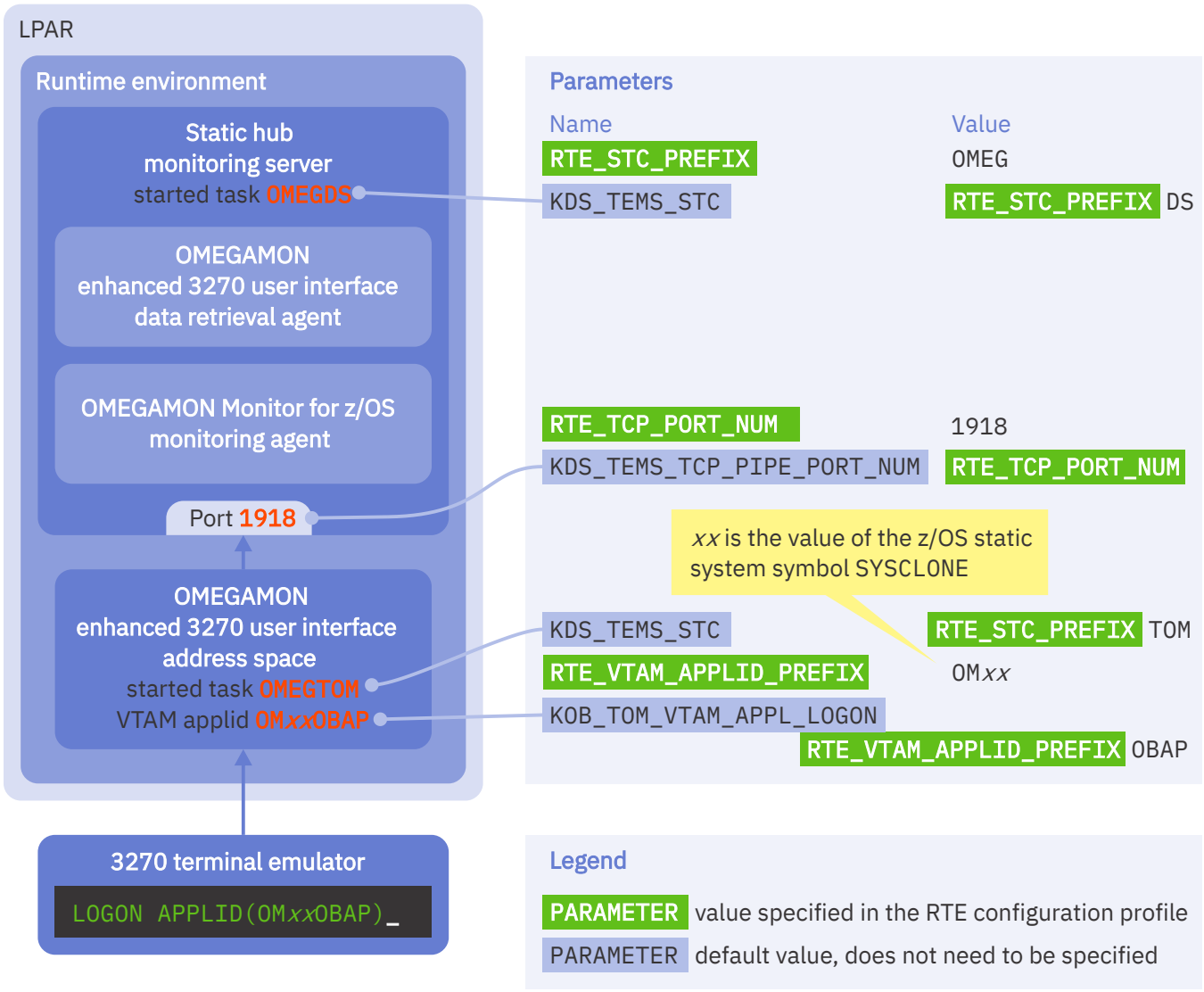

Figure 6. Parameters that specify z/OS-related identifiers for a minimal runtime environment

3. Submit a job that performs the **DISCOVER** action.

   The **DISCOVER** action discovers CICS regions, Db2 subsystems, IMS control regions, MQ subsystems, TCP/IP stacks, and System Symbols, and then writes corresponding members to the RTEDEF library.

   Strictly speaking, for this minimal runtime environment, you can skip this step, because this runtime environment will run only the z/OS agent, which does not require any subsystem parameters. However, performing the **DISCOVER** action is still a useful exercise, because it prepares the RTEDEF library for extending the runtime environment to run other agents.

   Reuse the same JCL as before, with the following changes:

   1

   Optionally, change the program name in the JCL **EXEC** statement to KCIALPHA.

   KCIALPHA is an APF-authorized version of KCIOMEGA. APF-authorization enables the program to discover more subsystem details.

   2

   Change the action to **DISCOVER**.

   Example JCL:

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//S1       EXEC PGM=KCIALPHA,REGION=0M,DYNAMNBR=256  1
//STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            DISCOVER  2
RTE_NAME          <rte_name>
RTE_PLIB_HILEV    <rte_plib_hilev>
/*
```

*Figure 7. Example JCL to perform the **DISCOVER** action*

4. Submit a job that performs the **GENERATE** action.

   Reuse the same JCL as before, with the following changes:

   1
   If you changed the program name to KCIALPHA for the **DISCOVER** action, change it back to KCIOMEGA before performing the **GENERATE** action.

   2
   Change the action to **GENERATE**.

   Example JCL:

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256  1
//STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            GENERATE  2
RTE_NAME          <rte_name>
RTE_PLIB_HILEV    <rte_plib_hilev>
/*
```

*Figure 8. Example JCL to perform the **GENERATE** action*

   The **GENERATE** action generates the runtime members for the runtime environment, including the started tasks.

   You have completed the steps that involve the configuration software, Monitoring Configuration Manager.

   The remaining steps complete the configuration of the runtime environment *outside* of the configuration software.

   These "complete the configuration" steps depend on your site-specific procedures and the requirements of the components, such as the monitoring agents, that you have chosen to configure in the runtime environment. The requirements of each component are described in the separate product documentation for each component.

   Typically, at this point in the procedure for creating a runtime environment, you would need to refer to that separate documentation. However, to help make this "first runtime environment" procedure stand-alone, and because in this procedure we have selected a fixed set of specific components, the "complete the configuration" steps are presented here.

   Depending on your user privileges, you might need to ask someone else to perform some or all of the following steps. For example, only z/OS system administrators are typically allowed to write to system libraries.

5. Use your site-specific procedures to copy the runtime members for started tasks and VTAM definitions to your system libraries.

   Copy the members from the following libraries to your corresponding PROCLIB, VTAMLIB, and VTAMLST system libraries:

   rte_hilev.SYS1.PROCLIB
   rte_hilev.SYS1.VTAMLIB

rte_hilev.SYS1.VTAMLST

The default value of the RTE_HILEV parameter is the value of RTE_PLIB_HILEV.

If you followed the earlier tip to set the **RTE_X_STC_INAPF_INCLUDE_FLAG** parameter to Y, then you can skip the next two steps. However, you should still *read* these steps to understand the requirements of the runtime environment for VTAM resources and APF-authorized libraries.

6. Activate the VTAM resources defined by this runtime environment.

   Issue the following **VARY ACT** MVS system command:

   ```
   VARY NET,ACT,ID=rte_vtam_applid_prefixNODE,SCOPE=ALL
   ```

   The **ID** parameter of the **VARY ACT** command must match the value of the runtime environment parameter **RTE_VTAM_GBL_MAJOR_NODE**.

   The default value of **RTE_VTAM_GBL_MAJOR_NODE** is the value of **RTE_VTAM_APPLID_PREFIX** followed by the string NODE. If you use the **RTE_VTAM_APPLID_PREFIX** initial value of OM*xx*, then the default value of **RTE_VTAM_GBL_MAJOR_NODE** is OM*xx*NODE, where *xx* is the value of the z/OS static system symbol **SYSCLONE**. For example, if the system (LPAR) name is Z0S1, then specify ID=OMS1NODE.

7. APF-authorize libraries.

   Add the following data sets to the authorized program facility (APF) list:

   - The following runtime environment library:

     rte_hilev.*rte_name*.RKANMODU

   - The following target libraries, under the high-level qualifiers of the STEPLIB of the Monitoring Configuration Manager job:

     TKANMOD
     TKANMODL
     TKANMODP
     TKANMODR

   The runtime member *rte_hilev*.SYS1.PROCLIB(*rte_stc_prefix*APF) contains **VARY ACT** and **SETPROG APF** commands for this runtime environment. Different runtime environments require different VTAM resources and APF-authorized libraries, depending on the configured products.

   If you specify the parameter **RTE_X_STC_INAPF_INCLUDE_FLAG** Y in the RTEDEF(*rte_name*) member, and then perform the **GENERATE** action, some started tasks will contain an **INCLUDE** statement to include that member, so that you do not need to issue these commands separately. Whether started tasks are allowed to perform such commands depends on your local site practices.

   Setting **RTE_X_STC_INAPF_INCLUDE_FLAG** can be expedient for initial testing. However, typical best practice is to activate VTAM resources and APF-authorize libraries during system initialization rather than each time you start a task. Use the generated SYS1.PROCLIB(*rte_stc_prefix*APF) member to identify which libraries you need to add to the APF list at system initialization.

8. Start at least the following tasks: *rte_stc_prefix*CN, *rte_stc_prefix*DS, and *rte_stc_prefix*TOM.

   The user ID that you associate with these started tasks must have z/OS UNIX superuser privileges and access to the runtime members.

   ***rte_stc_prefix*CN**
   OMEGAMON subsystem.

   The OMEGAMON subsystem does not belong to a runtime environment. You only need one OMEGAMON subsystem per LPAR.

   If the OMEGAMON subsystem has already been started, the job for this started task will fail. The JESMSGLG output data set for the failed job will contain the following messages:

```
CNDL018I  OMEGAMON SUBSYSTEM ALREADY ACTIVE ...
CNDL002I  OMEGAMON SUBSYSTEM V vrm TERMINATED ...
```

This is not a problem: your runtime environment will use the already-active subsystem.

***rte_stc_prefix*DS**
Monitoring server.

***rte_stc_prefix*TOM**
Enhanced 3270 user interface.

What to do next

Logon to the enhanced 3270 user interface and view the monitoring data from the z/OS agent.

Related concepts

Troubleshooting
Use these topics to troubleshoot issues with Monitoring Configuration Manager.

Preparing to use Configuration Manager
Review prerequisites and other considerations before you start using Configuration Manager.

Creating or updating a runtime environment

To create or update a runtime environment, you edit a set of parameters, and then you submit a job that performs the **GENERATE** action to generate runtime members from those parameters.

Before you begin

Read the prerequisites for using Monitoring Configuration Manager.

If you are updating a runtime environment after using SMP/E to apply maintenance to the target libraries, and your runtime environment refers to a copy of those libraries, then use your site-specific procedures to refresh that copy.

About this task

When creating a runtime environment, you can optionally perform the **CREATE** action to create an initial set of parameters.

When creating or updating a runtime environment, you can optionally perform the **DISCOVER** action to create or update subsystem parameters, rather than editing them yourself.

Updating a runtime environment encompasses many scenarios. For example:

- Applying maintenance, after using SMP/E to update target libraries
- Upgrading an agent to a new product release
- Adding or removing agents
- Changing parameter values; for example, to change a hub monitoring server to a remote monitoring server

The procedure for updating the runtime environment is the same for every scenario.

Procedure

1. Create a runtime environment definition. If you are updating a runtime environment, the definition already exists: skip this step.

   A runtime environment definition consists of one or more plain-text members in a library. These members specify the parameters that define the runtime environment. For details, see “Runtime environment definition (RTEDEF) library” on page 107.

   To create a runtime environment definition, submit a job that performs the **CREATE** action of Monitoring Configuration Manager.

   Example JCL:

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            CREATE
RTE_NAME          <rte_name>
RTE_PLIB_HILEV    <rte_plib_hilev>
/*
```

   *Figure 9. Example JCL to perform the **CREATE** action*

   Similar JCL is supplied in the KFJJMCM member of the TKANSAM target library.

   Edit the example job statement to match your site's standards. For example, for job name, class, and message class.

Replace the placeholders in the example JCL with appropriate values. For details, see "Batch interface" on page 31.

Tip: After running a Monitoring Configuration Manager job, check the KCIPRINT sysout data set.

2. Edit the parameters in the RTEDEF library to meet your requirements for the runtime environment.
3. Optionally, submit a job that performs the **DISCOVER** action.

   The **DISCOVER** action discovers CICS, Db2, IMS, and MQ subsystems, TCP/IP stacks, and System symbols, and then writes corresponding members to the RTEDEF library.

   Use the same JCL as the previous step but change the action to **DISCOVER**.

   Optionally, change the program name in the JCL **EXEC** statement to KCIALPHA. KCIALPHA is an APF-authorized version of KCIOMEGA. APF-authorization enables the program to discover more subsystem details.

   Check and, if necessary, edit the contents of the members created by the **DISCOVER** action.

   For discovered Db2 subsystems, you need to complete some of the parameters: see "Completing the parameters for discovered Db2 subsystems" on page 23.
4. Submit a job that performs the **GENERATE** action.

   Use the same JCL shown in the first step but change the action to **GENERATE**.

   The **GENERATE** action generates the runtime members for the runtime environment, including the started tasks.

   Note: You have completed the steps that involve the configuration software, Monitoring Configuration Manager.

   The remaining steps complete the configuration of the runtime environment *outside* of the configuration software.

   These "complete the configuration" steps depend on your site-specific procedures and the requirements of the components, such as the monitoring agents, that you have chosen to configure in the runtime environment. The requirements of each component are described in the separate product documentation for each component.

   Depending on your user privileges, you might need to ask someone else to perform some or all of the following steps. For example, only z/OS system administrators are typically allowed to write to system libraries.
5. Use your site-specific procedures to copy the runtime members for started tasks and VTAM definitions to your system libraries.

   Copy the members from the following libraries to your corresponding PROCLIB, VTAMLIB, and VTAMLST system libraries:

   rte_hilev.SYS1.PROCLIB
   rte_hilev.SYS1.VTAMLIB
   rte_hilev.SYS1.VTAMLST

   The default value of the RTE_HILEV parameter is the value of RTE_PLIB_HILEV.
6. Follow the instructions in the OMEGAMON shared documentation to complete the configuration of the runtime environment.
7. For a newly created runtime environment: start the tasks. For an updated runtime environment: stop and then restart the tasks.

Related concepts

Troubleshooting
Use these topics to troubleshoot issues with Monitoring Configuration Manager.

Preparing to use Configuration Manager

Review prerequisites and other considerations before you start using Configuration Manager.

Sharing runtime members with an SMP/E target installation library or creating a full, stand-alone set of runtime members

Runtime members can be either a full stand-alone set or they can refer to some read-only members, such as load modules, in SMP/E target installation libraries.

Procedure

1. In the RTEDEF(*rte_name*) member, specify an **RTE_TYPE** value.

   Valid values:

   FULL
   Stand-alone runtime members. Runtime members have no dependency on target libraries.

   SHARING
   Some runtime members refer to the target libraries.

   The high-level qualifiers of the target libraries are specified by the **GBL_TARGET_HILEV** parameter.

   SHARING reduces the storage requirement for each runtime environment.

   If **RTE_TYPE** is SHARING, then the value of the RTE_SHARE parameter must be SMP.

   If you omit **RTE_TYPE**, the default value is SHARING.
2. Submit a job that performs the **GENERATE** action.

Completing the parameters for discovered Db2 subsystems

The **DISCOVER** action discovers only some of the parameter values required to monitor Db2 subsystems. You must supply the remaining values.

Before you begin

You must have performed a **DISCOVER** action that created an RTEDEF(KD5@*lpar*) member containing a table of parameters: one row for each discovered Db2 subsystem.

About this task

You must supply values for the following parameters:

KD2_DBnn_DB2_RUNLIB
The Db2 RUNLIB library.

KD2_DBnn_DB2_PORT_NUM
The port number on which the OMEGAMON for Db2 Collector (or "server", default started task suffix 02) listens for requests.

The **DISCOVER** action sets a placeholder value for the port number. Typically, you will need to change this value to match your site-specific standards.

To specify the RUNLIB library for discovered Db2 subsystems, you must either ensure that the global parameters are correct or edit the parameters for each Db2 subsystem.

Procedure

1. Specify the correct port numbers.

   In the RTEDEF(KD5@*lpar*) member, change the placeholder value of each **KD2_DBnn_DB2_PORT_NUM** parameter to the actual port number you want to use.

2. Specify Db2 RUNLIB libraries.

   Select one of the following choices:

   - To use global parameter values for Db2 LOADLIB and RUNLIB libraries: in the RTEDEF(GBL$PARM) member or LPAR-specific RTEDEF(GBL$*lpar*) member, define **GBL_DSN_DB2_LOADLIB_Vn** and **GBL_DSN_DB2_RUNLIB_Vn** parameters for the Db2 versions that your site uses.
   - To set LOADLIB and RUNLIB libraries for each Db2 subsystem: in the RTEDEF(KD5@*lpar*) member, specify a value for **KD2_DBnn_DB2_RUNLIB**.

Related tasks

DISCOVER

The **DISCOVER** action discovers CICS regions, Db2 subsystems, IMS control regions, MQ subsystems, TCP/IP stacks and system symbols on an LPAR, and then creates corresponding members in the runtime environment definition library.

Related reference

RTEDEF(KD5@lpar)

If the **DISCOVER** action discovers Db2 subsystems, it creates the RTEDEF(KD5@*lpar*) member. This member contains parameters that configure the Db2 monitoring agent.

Converting a hub monitoring server to a remote monitoring server

Initially, you might configure a new runtime environment to be stand-alone, with its own hub monitoring server. Later, you can integrate that runtime environment with the rest of your monitoring topology by converting its hub monitoring server to a remote monitoring server that communicates with a central hub.

Before you begin

The following procedure assumes that you have already created the following two runtime environments:

- A runtime environment with a hub monitoring server that you want to convert to a remote monitoring server.
- A runtime environment with the central hub monitoring server that you want the new remote monitoring server to communicate with. We'll call this the **central hub** runtime environment.

About this task

Converting a hub monitoring server to a remote monitoring server involves changing the **KDS_TEMS_TYPE** parameter value from HUB to REMOTE, and setting some other **KDS_*** parameters to refer to the central hub.

The following diagram shows a stand-alone runtime environment with a hub:

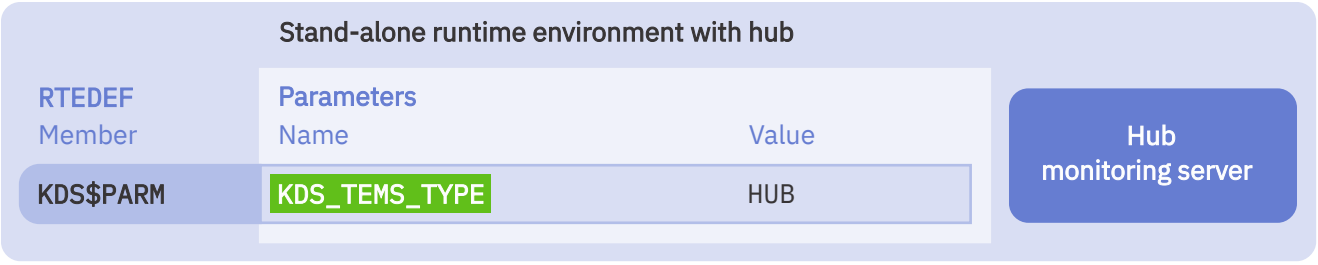

Figure 10. Before: A stand-alone runtime environment with a hub monitoring server

The following diagram shows the runtime environment after you have converted its hub to a remote monitoring server that communicates with a central hub:

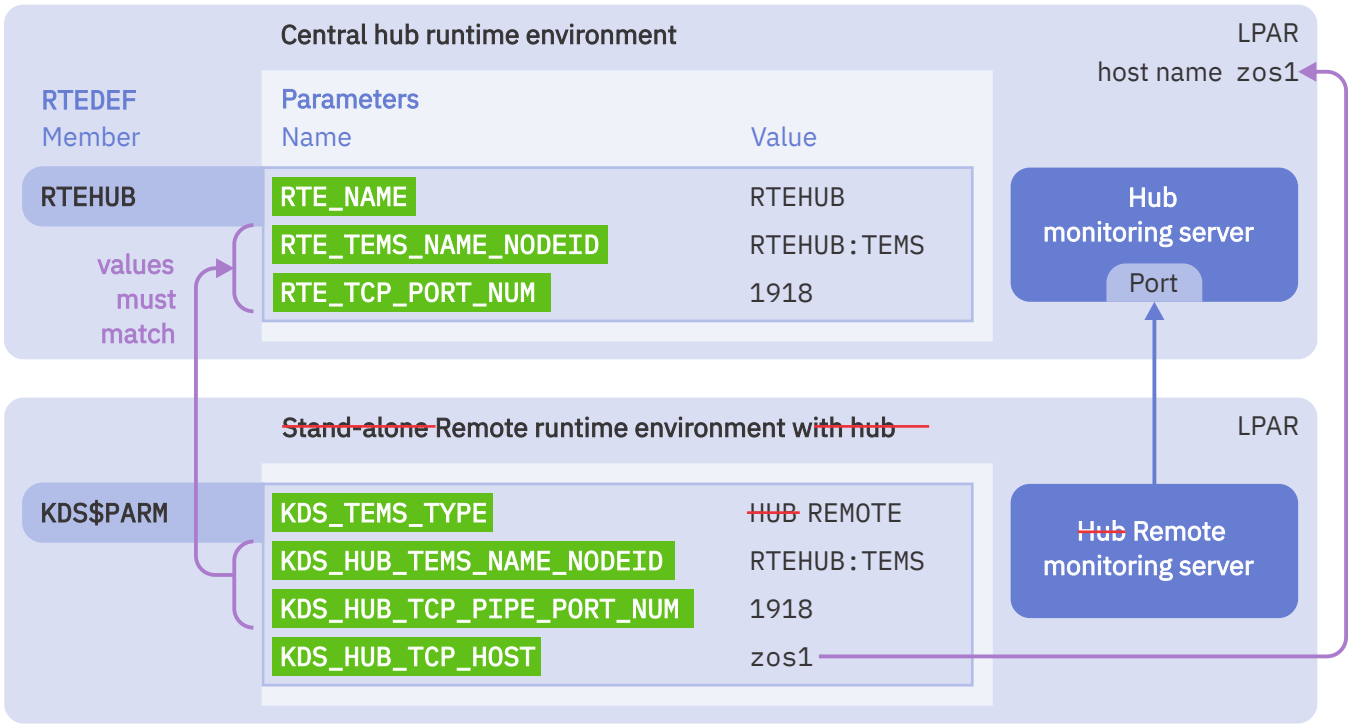

Figure 11. After: A runtime environment with a remote monitoring server

Procedure

1. Edit the RTEDEF(KDS$PARM) or RTEDEF(KDS$*lpar*) member for the runtime environment with the hub that you want to convert to a remote monitoring server.

   Set the following parameters:

   KDS_TEMS_TYPE
   Change from HUB to REMOTE.

   KDS_HUB_TEMS_NAME_NODEID
   Set to the value of **RTE_TEMS_NAME_NODEID** in the central hub runtime environment.

   KDS_HUB_TCP_PIPE_PORT_NUM
   Set to the value of **RTE_TCP_PORT_NUM** in the central hub runtime environment.

   KDS_HUB_TCP_HOST
   Set to the host name of the LPAR for the central hub runtime environment.
2. Submit a job that performs the **GENERATE** action for the runtime environment whose parameters you have just edited.

Related reference

Communication between monitoring components

In a typical topology, monitoring agents communicate with remote monitoring servers, and remote monitoring servers communicate with a single, central hub monitoring server.

Defining multiple runtime environments in an RTEDEF library

You can define one runtime environment per RTEDEF library or, as described here, you can define multiple runtime environments in a single RTEDEF library.

About this task

For each runtime environment that you want to define, you create a corresponding RTEDEF (*rte_name*) member. To cater for LPAR-specific parameter differences between runtime environments, you create LPAR-specific RTEDEF members.

The simple example presented here defines two runtime environments on different LPARs:

- A runtime environment with a hub monitoring server and an enhanced 3270 user interface, but no monitoring agents.
- A runtime environment with a remote monitoring server and a z/OS monitoring agent.

The following diagram shows the significant RTEDEF members and their parameters for this example:

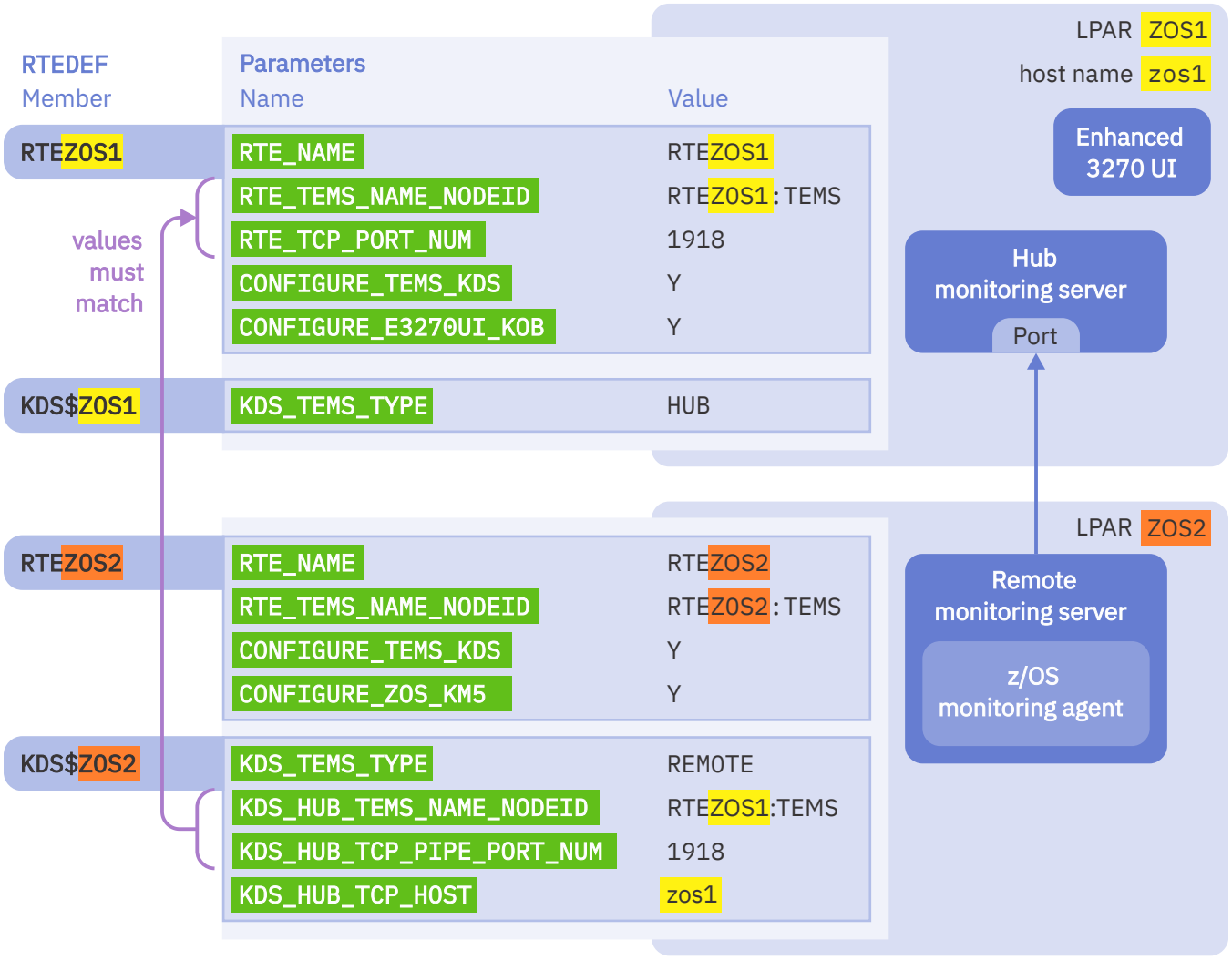

Figure 12. Defining two runtime environments in a single RTEDEF library

In the following procedure, replace the example names with names appropriate for your site:

- Replace the host name zos1 with the host name of the LPAR where you will run the hub monitoring server.
- Replace the LPAR names Z0S1 and Z0S2 with the names of LPARs at your site.
- Use RTE names that match your site naming conventions.

Procedure

1. Use the **CREATE** action to create an initial set of parameters in a new RTEDEF library.

   Example JCL:

   ```
   //UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
   /*JOBPARM SYSAFF=ZOS1 1
   //S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
   //STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
   //KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
   //KCIVARS  DD *
   ACTION            CREATE
   RTE_NAME          RTEZOS1 2
   RTE_PLIB_HILEV    TSOUID.MONSUITE
   /*
   ```

   1
   Run this job on the LPAR where you will run the hub monitoring server, ZOS1.

   2
   Specify the **RTE_NAME** that you want to use for the runtime environment with the hub monitoring server. For example, RTEZ0S1: the prefix RTE, followed by the LPAR name. You are free to use your own naming convention: for example, there is no requirement for the value to begin with RTE or to end with the LPAR name.
2. Edit the RTEDEF(RTEZ0S1) member to configure a monitoring server and an enhanced 3270 user interface.

   Set the following parameters to Y:

   CONFIGURE_TEMS_KDS to configure a monitoring server
   CONFIGURE_E3270UI_KOB to configure an enhanced 3270 user interface

   Either delete all other **CONFIGURE_*** parameters or set them to N.

   Review the following parameter values and, if necessary, change them to match your site requirements:

   RTE_TEMS_NAME_NODEID
   RTE_TCP_PORT_NUM
3. Create an RTEDEF(KDS$Z0S1) member.

   Set a single parameter in this member:

   KDS_TEMS_TYPE HUB

   Strictly speaking, this member is unnecessary, because HUB is the default value of **KDS_TEMS_TYPE**. However, creating this member serves as a reminder that the runtime environment on this LPAR contains a *hub* monitoring server. You will also need this member if you decide later to further configure the hub: for example, to make it a high-availability hub (**KDS_TEMS_HA_TYPE** HA).

   You have completed the definition of the runtime environment for the hub.
4. Run another job that performs a **CREATE** action, this time for the LPAR where you will run the remote monitoring server.

   Example JCL:

   ```
   //UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
   /*JOBPARM SYSAFF=ZOS2 1
   //S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
   //STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
   //KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
   ```

```
//KCIVARS  DD *
ACTION            CREATE
RTE_NAME          RTEZOS2  2
RTE_PLIB_HILEV    TSOUID.MONSUITE
/*
```

1
Run this job on the LPAR where you will run the remote monitoring server, ZOS2.

2
Specify the **RTE_NAME** that you want to use for the runtime environment with the remote monitoring server. For example, RTEZ0S2.

5. Edit the new RTEDEF(RTEZ0S2) member to configure a monitoring server and a z/OS monitoring agent.

   Set the following parameters to Y:

   CONFIGURE_TEMS_KDS to configure a monitoring server
   CONFIGURE_ZOS_KM5 to configure a z/OS monitoring agent

   Either delete other **CONFIGURE_*** parameters or set them to N.
6. Create an RTEDEF(KDS$ZOS2) member.

   Set the following parameters in the member:

   KDS_TEMS_TYPE
   Set to REMOTE.

   KDS_HUB_TEMS_NAME_NODEID
   Set to the value of **RTE_TEMS_NAME_NODEID** in the hub runtime environment.

   KDS_HUB_TCP_PIPE_PORT_NUM
   Set to the value of **RTE_TCP_PORT_NUM** in the hub runtime environment.

   KDS_HUB_TCP_HOST
   Set to the host name of the LPAR for the hub runtime environment.
7. Optionally, delete the RTEDEF(KDS$PARM) member.

   The KDS$PARM member created by the **CREATE** action contains only one parameter, **KDS_TEMS_TYPE**. The value of **KDS_TEMS_TYPE** in the LPAR-specific members takes precedence over the value in KDS$PARM. So the KDS$PARM member is, effectively, redundant.

   You have completed the definition of the runtime environment for the remote monitoring server.
8. Use the **GENERATE** action to create runtime members for the hub runtime environment.

   Example JCL:

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1  1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            GENERATE
RTE_NAME          RTEZOS1  2
RTE_PLIB_HILEV    TSOUID.MONSUITE
/*
```

1
Run this job on the LPAR where you will run the hub, ZOS1.

2
Specify an **RTE_NAME** value to match the RTEDEF(RTEZ0S1) member for the hub runtime environment.

9. Use the **GENERATE** action to create runtime members for the remote monitoring server runtime environment.

   Reuse the JCL from the previous step, with the following changes:

1. Run the job on the LPAR where you will run the remote monitoring server, ZOS2.
2. Specify the corresponding **RTE_NAME** for that runtime environment, RTEZ0S2.

What to do next

For details on completing the configuration of these runtime environments and then starting them, see the corresponding steps in the general procedure for creating a runtime environment.

Extend this example with more LPARs and more monitoring agents. Use the **DISCOVER** action to discover subsystems on each LPAR.

This example assumed that the same global parameter values in RTEDEF(GBL$PARM) apply to the runtime environments on both LPARs. To specify different values for different LPARs, create LPAR-specific RTEDEF(GBL$*lpar*) members.

Related reference

Runtime environment definition library members

RTEDEF library members follow a naming convention that identifies the contents of the member and whether the member applies to a specific LPAR or to all LPARs.

Batch interface

The JCL to run IBM Z Monitoring Configuration Manager is simple and concise. You specify an action, the name of the runtime environment on which you want to perform that action, and the high-level qualifiers of the data sets for that runtime environment.

JCL

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
*
*
*<action> SELECT FROM:
* CREATE|DISCOVER|GENERATE|DELETE|MIGRATE|PACKAGE|DEPLOY
*
ACTION            <action>
OPTION            <option>
RTE_NAME          <rte_name>
RTE_PLIB_HILEV    <rte_plib_hilev>
/*
```

Figure 13. JCL to run Monitoring Configuration Manager

Similar JCL is supplied in the KFJJMCM member of the TKANSAM target library.

KCIOMEGA is the program that runs Monitoring Configuration Manager. The KCIFLOW data set provides input to KCIOMEGA. That is all you need to know about the KCIOMEGA program and the KCIFLOW data set to run Monitoring Configuration Manager. If you want to know more, see "KCIOMEGA workflows" on page 66.

Replace the placeholders in the JCL with appropriate values:

<lpar>
Run Monitoring Configuration Manager actions on the LPAR where you will start the runtime environment.

For example, if your site uses JES2, insert a **SYSAFF** job parameter after the **JOB** statement to ensure that the job runs on that LPAR.

<tlib_hlq>
The high-level qualifiers of the target libraries.

<action>
One of the Monitoring Configuration Manager actions.

You can abbreviate actions to their first three characters: **CRE**, **DIS**, **GEN**, **DEL**, **MIG**, **PAC**, and **DEP**.

<option>
One or more compatible Monitoring Configuration Manager options. Multiple options must be specified by a comma with no spaces.

<rte_name>
Runtime environment name, 1 - 8 characters.

Monitoring Configuration Manager uses this name for various purposes, including:

- MVS member names
- MVS data set name qualifiers
- z/OS UNIX System Services directory name, all uppercase

<rte_plib_hilev>
The high-level qualifiers of the runtime environment definition library:

rte_plib_hilev.RTEDEF

Monitoring Configuration Manager uses the values of **RTE_NAME** and **RTE_PLIB_HILEV** to set the default value of other parameters, such as **RTE_HILEV** and **RTE_VSAM_HILEV**, that are used for data set names. To avoid exceeding the z/OS 44-character limit for data set names, the combined length of **RTE_NAME** and **RTE_PLIB_HILEV** should not exceed 28 characters. For example, if **RTE_NAME** is 8 characters, then **RTE_PLIB_HILEV** should not exceed 20 characters.

RTE_NAME and RTE_PLIB_HILEV parameters versus the values in KCIVARS

The **CREATE** action uses the **RTE_NAME** and **RTE_PLIB_HILEV** values that you specify in the KCIVARS input data set as the initial values of the **RTE_NAME** and **RTE_PLIB_HILEV** parameters in the *rte_plib_hilev*.RTEDEF(*rte_name*) member. At that point in time, the values of the **RTE_NAME** and **RTE_PLIB_HILEV** parameters in the RTEDEF(*rte_name*) member match the values that you specified in KCIVARS.

However, you might edit the values of the **RTE_NAME** and **RTE_PLIB_HILEV** parameters in the RTEDEF(*rte_name*) member so they no longer match the values that you specified in KCIVARS.

For subsequent actions, the **RTE_NAME** and **RTE_PLIB_HILEV** values that you specify in KCIVARS are used only to *locate* the *rte_plib_hilev*.RTEDEF(*rte_name*) member. The action uses the **RTE_NAME** and **RTE_PLIB_HILEV** parameters in that RTEDEF library.

Contents of the KCIVARS input data set

The contents of the KCIVARS data set are **case-sensitive**: you must specify the variable names and their values exactly as described.

KCIVARS can contain comment lines and inline comments:

- Comment lines begin with an asterisk (*) in column 1.

```
* Comment line
```

- Inline comments begin with an asterisk after a variable value.

```
RTE_NAME MYRTE * Inline comment
```

Actions

IBM Z Monitoring Configuration Manager can perform several actions. The main action, **GENERATE**, generates runtime members, which is the main purpose of Monitoring Configuration Manager. The other actions are optional, for your convenience.

You typically perform the actions in the following order:

1. **CREATE**
2. **DISCOVER**
3. **GENERATE**
4. **DELETE**

In addition, a **MIGRATE** action can be used to copy the required configuration parameters from PARMGEN for use with Monitoring Configuration Manager. When using **MIGRATE**, you typically perform the actions in the following order:

1. **MIGRATE**
2. **GENERATE**
3. **DELETE**

Finally, to support cross-sysplex rollout scenarios, the **PACKAGE** and **DEPLOY** actions can be used after the **GENERATE** action has completed successfully.

Action options

The **OPTION** parameter is available for some actions and provides more granular control over certain processing. Use of this parameter is optional.

This topic summarizes the available options for each of the respective actions, as follows. For sample JCL, see "JCL" on page 31.

CREATE action

The following options are available for the **CREATE** action. For more information, see "CREATE" on page 34.

MULTIPLE
Use a single Configuration Manager RTEDEF data set for multiple runtime environments. You can abbreviate this keyword to **MULTI**.

TRGCOPY
Create a target copy member in the RTEDEF library. You can abbreviate this keyword to **TRG**.

GENERATE action

Some options require a complete run of the **GENERATE** action prior to use. Additionally, some options are not compatible to run with other options during the same job.

To specify more than one option, separate the values with a comma and no spaces. For example: **OPTION USS,SECEXITS**

The following options are available for the **GENERATE** action. For details about using these options, see "GENERATE" on page 45.

USS
Run only the **GENERATE** workflow stage that deploys the parts related to z/OS UNIX System Services. This option requires a complete run of the **GENERATE** action prior to use.

NOUSS
Do not run the z/OS UNIX deploy stage in the **GENERATE** action.

SECEXITS
Perform configuration for security exits only. This option requires a complete run of the **GENERATE** action prior to use. You can abbreviate this keyword to **SEC**.

VALIDATE
Perform initial validation of RTEDEF parameters. You can abbreviate this keyword to **VAL**.

QUICKLOAD
Load the read-only configuration members to the RK* data sets. The read-only members are those members that are not impacted by customization during configuration. You can abbreviate this keyword to **QL**.

QUICKCONFIG
Update the configurable members for the runtime environment (for example, in the RKANPARU, RKANSAMU, and RKANCMDU libraries) without refreshing data from SMP/E target libraries. You can abbreviate this keyword to **QC**.

RELINK
Assemble and link edit modules for OMEGAMON for Networks (KN3) and OMEGAMON enhanced 3270 user interface (KOB). You can abbreviate this keyword to **LINK**.

TRGCOPY
Copy SMP/E target libraries into target copy data sets. You can abbreviate this keyword to **TRG**.

MIGRATE action

The following option is available for the **MIGRATE** action. For more information, see "MIGRATE" on page 55.

MULTIPLE
Use a single Configuration Manager RTEDEF data set for multiple PARMGEN runtime environments. You can abbreviate this keyword to **MULTI**.

PACKAGE action

The following option is available for the **PACKAGE** action. For more information, see "PACKAGE" on page 59.

NOUSS
Do not include the files and directories related to z/OS UNIX in the **PACKAGE** output.

DEPLOY action

The following options are available for the **DEPLOY** action. For more information, see "DEPLOY" on page 61.

USS
Run only the **DEPLOY** workflow stage that deploys the parts related to z/OS UNIX.

NOUSS
Do not run the z/OS UNIX deploy stage in the **DEPLOY** action.

All actions

The following option is available for all actions.

DEBUG
This option provides output for troubleshooting purposes and should only be used under the guidance of IBM Software Support. The **DEBUG** keyword can be used with all other **OPTION** keywords. You can abbreviate this keyword to **DBG**.

CREATE

The **CREATE** action creates an initial runtime environment definition that you can customize to match your requirements.

Before you begin

Review the following information before you use the **CREATE** action:

- For an overview of how the **CREATE** action fits into the process of creating a runtime environment, see "Creating your first, minimal runtime environment" on page 13.
- For information about the members that the **CREATE** action allocates and populates, see "Initial runtime environment library members" on page 110.
- The **CREATE** action supports creating one or more runtime environments in a single RTEDEF configuration. It is recommended that you decide prior to the creation of your first runtime environment whether you plan to create one or multiple runtime environments in a single RTEDEF.

  Note: If you are going to set up a High Availability TEMS (HA TEMS), make sure only one runtime environment is defined in the RTEDEF (that is, the one used for the HA TEMS).

Note: You can also use the **CREATE** action as part of the process to create a copy of your SMP/E target libraries. For more information, see "Using SMP/E target library copies" on page 141.

About this task

The **CREATE** action creates an initial runtime environment definition by allocating and populating the necessary data sets, members, and configuration settings.

Note: Using the **CREATE** action to create your initial runtime environment definition is optional and provided for convenience; you can perform the same steps manually. Experienced users can skip **CREATE** and copy an existing RTEDEF library. You can allocate the RTEDEF library yourself using a record format of fixed-length, blocked (FB) and a record length of 80. You can also create the members; the only required member is *rte_name*.

The following list provides details about the **CREATE** action:

- The **CREATE** action allocates the runtime environment definition library, *rte_plib_hilev*.RTEDEF, if it does not already exist, and populates it with initial configuration settings. The **CREATE** action does not overwrite members. If the RTEDEF library already exists, the **CREATE**action only writes members that do not yet exist.
- You can create one or more runtime environments in a single RTEDEF configuration. The default behavior of the **CREATE** action is to create only one runtime environment in the RTEDEF data set. Using the **OPTION MULTIPLE** parameter, you can create multiple runtime environments in a single RTEDEF data set. Each runtime environment creation requires a separate **CREATE** action job. If you plan to configure multiple runtime environments in a single RTEDEF data set, make sure to include the **OPTION MULTIPLE** parameter on every **CREATE** action job, including the first one.

  Note: You can abbreviate **OPTION MULTIPLE** to **OPTION MULTI**. **OPTION MULTIPLE** is not compatible with the other available **CREATE** action option, **TRGCOPY**.
- The **CREATE** action creates the necessary members in the RTEDEF data set, as follows:
  - When using the default behavior of the **CREATE** action to create one runtime environment in a single RTEDEF data set (omitting the **OPTION MULTIPLE** parameter), the **CREATE** action will create members of type K*pp*$PARM in the respective created RTEDEF data set, along with the *rte_name* member for the runtime environment-specific parameters.
  - When creating a runtime environment in a configuration where a single RTEDEF contains multiple runtime environments, use parameter **OPTION MULTIPLE** and **KFJ_SYSNAME lpar** in the KCIVARS DD. The **CREATE** action will create members of type K*pp*$*lpar* in the RTEDEF data set, along with the *rte_name* member for the runtime environment-specific parameters.

    Note: When using **OPTION MULTIPLE**, the **CREATE** action creates the K*pp*$*lpar* members automatically. It is recommended that you create the K*pp*$*lpar* members as well and add the parameter values that are the same for all runtime environments in the given RTEDEF library.

    On subsequent runs of the **CREATE** action, reuse the same **RTE_PLIB_HILEV** parameter value, but update the values for parameters **RTE_NAME** and **KFJ_SYSNAME** to create a new set of runtime environment parameter members. There is no limit on how many runtime environments can be configured in a single RTEDEF data set.
- The **CREATE** action allocates the security exits library with the default name *rte_plib_hilev*.*rte_name*.SECEXITS (or, optionally, the name specified in the **KFJ_SECURITY_EXITS_LIB** parameter). The **CREATE** action also populates the security exits library with default security exits members and defines the library to the runtime environment using the **RTE_X_SECURITY_EXIT_LIB** parameter. For more information, see "Setting up security exits in your runtime environment" on page 131.
- If the use of override embed members is enabled by specifying parameter **KFJ_USE_EMBEDS** set to Y, the **CREATE** action allocates the embeds data set with the default name *rte_plib_hilev*.*rte_name*.EMBEDS (or, optionally, the name specified in the **KFJ_EMBEDS_LIB** parameter). The **CREATE** action sets up the embeds data set, populates it with supported override embed parameters (if applicable), and defines it to the runtime environment using the **RTE_X_OVERRIDE_EMBEDS_LIB** parameter. For more information, see "Using override embed members" on page 133.

- The **CREATE** action works with the **KFJ_LOCAL_PLIB_HILEV** parameter to allow for local generation of runtime environments for remote systems using different high-level qualifiers.

  When the **KFJ_LOCAL_PLIB_HILEV** parameter is specified, the generated *kfj_local_plib_hilev*.RTEDEF data set will contain an additional member: PCK$PARM for a default (single) **CREATE** action, or member PCK$*lpar* in a multiple **CREATE** action. This member allows locally generated runtime environments using a different data set high-level qualifier than the one intended to be used on the deployment target (for example, the production system).

  For more information about remote deployments, see "Special considerations for SYSPLEX rollout" on page 111, "RTEDEF(PCK$PARM)" on page 114, and "Remote deployment scenario" on page 137.

To create a runtime environment definition using the **CREATE** action, use the following procedure.

Procedure

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a **CREATE** action.
2. Specify values for the required parameters **RTE_NAME** and **RTE_PLIB_HILEV**.
3. (Optional) Specify additional parameters as needed, for example:
   - To create this runtime environment in a RTEDEF data set that does or will contain multiple runtime environment configurations, add the **OPTION MULTIPLE** and **KFJ_SYSNAME lpar** parameters. If this is a subsequent run of the **CREATE** action, reuse the same **RTE_PLIB_HILEV** parameter value, but update the values for parameters **RTE_NAME** and **KFJ_SYSNAME**.
   - To specify a different name for the security exits library, add the **KFJ_SECURITY_EXITS_LIB** parameter and value.
   - To enable the use of override embed members, add the **KFJ_USE_EMBEDS** parameter set to Y and the **KFJ_EMBEDS_LIB** parameter and value.
4. Run the KFJJMCM job to generate and populate the RTEDEF data set and other required data sets.

   Job messages for the **CREATE** action are written to the KCIPRINT SYSOUT data set.

Example

Creating runtime environment definition for one LPAR

The following JCL jobs create the runtime environment definition library TSOUID.MONSUITE.RTEDEF and populate it with various members, including the runtime environment configuration profile member RTE1. The first example is for a single runtime environment RTEDEF, and the second example is for a multiple runtime environment RTEDEF. These examples also specify that override embed members are enabled and provide custom data set names for the security exits and embeds libraries.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
*
ACTION                  CREATE
RTE_NAME                RTE1
RTE_PLIB_HILEV          TSOUID.MONSUITE

KFJ_SECURITY_EXITS_LIB  TEST1.TST.DEMO.MYEXITS
KFJ_USE_EMBEDS          Y
KFJ_EMBEDS_LIB          TEST1.TST.DEMO.MYEMBEDS
/*
```

*Figure 14. Example JCL to perform the **CREATE** action (one LPAR) for a single runtime environment RTEDEF*

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
*
ACTION                  CREATE
OPTION                  MULTIPLE
RTE_NAME                RTE1
RTE_PLIB_HILEV          TSOUID.MONSUITE

KFJ_SECURITY_EXITS_LIB  TEST1.TST.DEMO.MYEXITS
KFJ_USE_EMBEDS          Y
KFJ_EMBEDS_LIB          TEST1.TST.DEMO.MYEMBEDS

KFJ_SYSNAME             &SYSNAME
/*
```

*Figure 15. Example JCL to perform the **CREATE** action (one LPAR) for a multiple runtime environment RTEDEF*

Creating a runtime environment definition for remote deployment

The following JCL creates the runtime environment definition library TSOUID.DEV.RTEDEF. In addition to the standard members, it will contain member PCK$PARM, where you will find all available **KFJ_LOCAL_*** parameters.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1 EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS DD *
*
ACTION                  CREATE
RTE_NAME                RTE1
RTE_PLIB_HILEV          TSOUID.PROD
KFJ_LOCAL_PLIB_HILEV    TSOUID.DEV
/*
```

*Figure 16. Example JCL to perform the **CREATE** action (remote deployment)*

Note: If **OPTION MULTIPLE** is used, the **CREATE** action creates member PCK$*lpar* instead of PCK$PARM.

Related tasks

PACKAGE
The **PACKAGE** action packages a runtime environment that can then be deployed to a remote system.

DEPLOY
The **DEPLOY** action deploys a packaged runtime environment to a remote system.

Related reference

Initial runtime environment library members
The **CREATE** action populates the RTEDEF library with an initial set of configuration profile members for a basic runtime environment.

Setting up security exits in your runtime environment
Security exits are required for your runtime environment. You can use the **CREATE** and **MIGRATE** actions to set up your security exits library, and use the **GENERATE** action to create the necessary runtime members.

Using override embed members

With Monitoring Configuration Manager, you can use override embed members to provide and maintain customization for your runtime environments.

DISCOVER

The **DISCOVER** action discovers CICS regions, Db2 subsystems, IMS control regions, MQ subsystems, TCP/IP stacks and system symbols on an LPAR, and then creates corresponding members in the runtime environment definition library.

Before you begin

This documentation uses *subsystem* as an informal collective term for CICS regions, Db2 subsystems, IMS control regions, MQ subsystems, and TCP/IP stacks.

Discovery is not limited to the products that you have configured using **CONFIGURE_*** parameters. The **DISCOVER** action always discovers all the subsystems it can.

It is recommended that you review the following topics, which describe the results of the **DISCOVER** action:

- "Parameters created by the DISCOVER action" on page 40
- "Members created by the DISCOVER action" on page 41

Tip: Bring up all the subsystems you intend to monitor before you run the **DISCOVER** action. You can re-run the **DISCOVER** action if any subsystems are added or started on your LPAR after your initial discovery run. In this case, you need to consolidate the respective discovery output in your RTEDEF data set (that is, merge the generated members accordingly).

About this task

The **DISCOVER** action discovers subsystems and system symbols on an LPAR, and then creates corresponding members in the runtime environment definition library (*rte_plib_hilev*.RTEDEF) and populates the members with the discovered data.

The following list provides details about the **DISCOVER** action:

- Run the **DISCOVER** action on the LPAR whose subsystems you want to discover. For example, insert a JES2 **SYSAFF** job parameter after the **JOB** statement to ensure that the job runs on the correct LPAR:

  ```
  /*JOBPARM SYSAFF=<lpar>
  ```

- Use the KCIALPHA program to discover more details. Optionally, for the **DISCOVER** action only, change the program name in the JCL **EXEC** statement from KCIOMEGA to KCIALPHA. KCIALPHA is an APF-authorized version of KCIOMEGA. APF authorization enables KCIALPHA to discover more subsystem details, which would otherwise need to be entered manually. If KCIALPHA is run from a load library that is not APF authorized, partial discovery will still be completed, but the job will end with return code 8.
- The **DISCOVER** action can run stand-alone, that is, without requiring an existing RTEDEF data set as pointed to by the **RTE_PLIB_HILEV** parameter. The resulting RTEDEF will only contain the members produced by discovery and will use the default prefixes for VTAM applids and started task definitions. For more information about this use case, see the example "Stand-alone discovery using defaults" on page 39.
- For some subsystem data sets, if an alias is defined for the data set, the **DISCOVER** action will use the alias name instead of the original data set name. The following parameters will be populated with alias names if they exist: KD2_DB*nn*_DB2_LOADLIB, KI2_I1*nn*_CLASSIC_IMS_RESLIB, KN3_TCPX*nn*_TCPIP_PROFILES_DSN.
- After performing a **DISCOVER** action, review the messages about discovery in the KCIPRINT SYSOUT data set.
- The output of the **DISCOVER** action depends on the subsystem. For more information, see the following topics:

  - "Parameters created by the DISCOVER action" on page 40
  - "Members created by the DISCOVER action" on page 41
- On the first run, the **DISCOVER** action creates members for each type of subsystem it discovers. On subsequent runs, if the discovery member for a subsystem exists, it will not be overwritten, but instead a comment member (K*pp*#*lpar*) will be created. You must then review the members and manually apply the updates that you want to keep. For more information, see "First-time discovery versus rediscovery" on page 42.

To run a discovery using the **DISCOVER** action, use the following procedure.

Procedure

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a **DISCOVER** action.
2. Specify values for the required parameters **RTE_NAME** and **RTE_PLIB_HILEV**.
3. Run the KFJJMCM job to discover the available subsystems.

   The **DISCOVER** action creates the discovery members in the RTEDEF data set and populates the parameters with discovered values. Job messages for the **DISCOVER** action are written to the KCIPRINT SYSOUT data set.
4. Review the content in the discovery members and update as necessary.

Example

Discovering subsystems

The following JCL discovers subsystems on the LPAR ZOS1 and creates corresponding members in TSOUID.MONSUITE.RTEDEF. For example, if the **DISCOVER** action discovers Db2 subsystems, then it creates the member KD5@ZOS1; or, if that member already exists, KD5#ZOS1.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIALPHA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            DISCOVER
RTE_NAME          RTE1
RTE_PLIB_HILEV    TSOUID.MONSUITE
/*
```

*Figure 17. Example JCL to perform the **DISCOVER** action*

The JES2 **SYSAFF** job parameter ensures that the job runs, and the **DISCOVER** action discovers subsystems, on LPAR ZOS1.

Stand-alone discovery using defaults

The following JCL is used to perform a stand-alone discovery that uses default values.

If you want to discover IMS systems using the stand-alone run of the **DISCOVER** action and specify different prefixes for started task and VTAM applids, specify the prefixes of your choice in the **RTE_VTAM_APPLID_PREFIX** and **RTE_STC_PREFIX** parameters. This way the VTAM and started task prefixes will be honored accordingly in member RTEDEF (*KI5@lpar*).

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1 EXEC PGM=KCIALPHA,REGION=0M,DYNAMNBR=256
//STEPLIB DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS DD *
ACTION                    DISCOVER
RTE_NAME                  RTE1
RTE_PLIB_HILEV            TSOUID.MONSUITE

RTE_STC_PREFIX            OMEG
RTE_VTAM_APPLID_PREFIX    OM&SYSCLONE.
/*
```

*Figure 18. Example JCL to perform a stand-alone discovery using the **DISCOVER** action*

If you do not have any SMP/E target libraries on the system on which you want to run a stand-alone **DISCOVER** action, you can use the utility, TKANSAM(KFJMAINT), with action **BLDREMDS** to build the necessary minimum data sets (TKANSAM, TKANMOD, and TKANCUS libraries) needed to run the action. Make sure you transfer the created data sets to your remote system where the stand-alone **DISCOVER** should run and where the necessary APF authorization of the TKANMOD library is made.

Related tasks

Completing the parameters for discovered Db2 subsystems
The **DISCOVER** action discovers only some of the parameter values required to monitor Db2 subsystems. You must supply the remaining values.

Parameters created by the DISCOVER action

The **DISCOVER** action creates parameters for each subsystem it discovers.

Overview of output

The output of the **DISCOVER** action depends on the subsystem:

*Table 3. Output of the **DISCOVER** action*

Subsystem	Output
Db2, IMS, TCP/IP	Parameters that configure the runtime environment to monitor these subsystems. For Db2, you need to complete the discovered details.
CICS	Parameters that specify an historical datastore allocation table for task history file disposition.
MQ	Statements using embed overrides that you can place in the KMQ$CUSR embed override member: `SET MANAGER NAME(queue-manager)`

In addition to the subsystem discovery, a member is created that contains the System symbols that can be used by the **GENERATE** action along with the respective **KCIPARSE** extracted variables. For more information, see “RTEDEF(SYS@lpar)” on page 45.

Parameters

The following table lists the parameters created by the **DISCOVER** action.

The parameters created depend on which program performs the action: KCIOMEGA or KCIALPHA.

In general, KCIOMEGA discovers only the subsystem identifier. For detailed discovery, use KCIALPHA.

*Table 4. Parameters created by the **DISCOVER** action*			
Subsystem	**Parameter (*nn* is the 2-digit table row number)**	**KCIOMEGA**	**KCIALPHA**
CICS	**KC2_HSnn_CLASSIC_CICS_REGION**	**No**	Yes
Db2	**KD2_DBnn_DB2_SSID**	Yes	Yes
Db2	**KD2_DBnn_DB2_VER**	**No**	Yes
Db2	**KD2_DBnn_DB2_DS_GROUP**	**No**	Yes
Db2	**KD2_DBnn_DB2_LOADLIB**	**No**	Yes
IMS	**KI2_I1nn_CLASSIC_IMSID**	Yes	Yes
IMS	**KI2_I1nn_CLASSIC_IMS_RESLIB**	**No**	Yes
MQ	For MQ subsystems, **DISCOVER** generates **SET MANAGER** statements, not parameters.	Yes	Yes
TCP/IP	**KN3_TCPXnn_TCP_STC**	Yes	Yes
TCP/IP	**KN3_TCPXnn_TCPIP_PROFILES_DSN**	**No**	Yes
TCP/IP	**KN3_TCPXnn_TCPIP_PROFILES_MBR**	**No**	Yes

Members created by the DISCOVER action

The **DISCOVER** action creates members for each type of subsystem it discovers.

*Table 5. Members created by the **DISCOVER** action*	
Subsystem	**Member name**
CICS	KC5@*lpar* for first-time discovery KC5#*lpar* for rediscovery
Db2	KD5@*lpar* for first-time discovery KD5#*lpar* for rediscovery
IMS	KI5@*lpar* for first-time discovery KI5#*lpar* for rediscovery
MQ	KMQ#*lpar* **Note:** For MQ subsystems, only the KMQ#*lpar* member is created for both first-time discovery and rediscovery. For more information, see “RTEDEF(KMQ#lpar)” on page 44.
TCP/IP	KN3@*lpar* for first-time discovery KN3#*lpar* for rediscovery
Symbols	SYS@*lpar* for first-time discovery SYS#*lpar* for rediscovery

Member naming convention

For subsystem and symbols discovery, the **DISCOVER** action creates RTEDEF library members with the following naming convention:

Prefix

For subsystem type: K*pp* from the corresponding **CONFIGURE_*** parameter.

For symbols: SYS.

Separator
An at sign (@) or a hash (#).

A hash indicates that the member is a comment: the **GENERATE** action ignores these members.

Note: The at sign (@) appears in only some code pages. For example, when using CCSID 1141, it appears as "§" (0x7C).

Suffix
Identifies the LPAR.

First-time discovery versus rediscovery

The **DISCOVER** action does not overwrite K*pp*@*lpar* or SYS@*lpar* members ("@" members).

If a K*pp*@*lpar* or SYS@*lpar* member already exists, the **DISCOVER** action writes a comment member, K*pp*#*lpar* or SYS#*lpar*, respectively, and then continues, eventually completing with return code 4. Review the comments about discovery in the KCIPRINT sysout data set, and then review the "#" members. Edit the existing "@" members to apply any preferred updates.

The **DISCOVER** action overwrites comment members.

Related reference
Initial runtime environment library members
The **CREATE** action populates the RTEDEF library with an initial set of configuration profile members for a basic runtime environment.

RTEDEF(KC5@lpar)

If the **DISCOVER** action discovers CICS regions, it creates the RTEDEF(KC5@*lpar*) member. This member contains parameters that configure the CICS monitoring agent.

```
* CICS regions discovered by KCIOMEGA
* SYSPLEX=sysplex LPAR=lpar DATE=date

KC2_HS                            BEGIN               * Table begin *

KC2_HS01_ROW                      01
KC2_HS01_CLASSIC_CICS_REGION      "region_name"
KC2_HS01_CLASSIC_VSAM_CYL         1  1

* More rows (one for each region discovered)...

KC2_HS                            END                 * Table end *
```

*Figure 19. RTEDEF(KC5@lpar) member created by the **DISCOVER** action*

1 KC2_HS01_CLASSIC_VSAM_CYL
If necessary, replace the initial value of 1 with an appropriate value for your site.

RTEDEF(KD5@lpar)

If the **DISCOVER** action discovers Db2 subsystems, it creates the RTEDEF(KD5@*lpar*) member. This member contains parameters that configure the Db2 monitoring agent.

```
* Db2 subsystems discovered by KCIOMEGA
* SYSPLEX=sysplex LPAR=lpar DATE=date

KD2_DB                      BEGIN                      * Table begin *

KD2_DB01_ROW                01
KD2_DB01_DB2_SSID           "ssid"
KD2_DB01_DB2_DESCRIPTION    "ssid Db2 subsystem"
KD2_DB01_DB2_VER            "version"
KD2_DB01_DB2_SYSNAME        "lpar"
KD2_DB01_DB2_PROFID         "P001"
KD2_DB01_DB2_DS_GROUP       "N"
KD2_DB01_DB2_MONITOR_START  "N"
KD2_DB01_DB2_PORT_NUM       "2000"
KD2_DB01_DB2_LOADLIB        "dsname"  1
KD2_DB01_DB2_DSNTIAD        "DSNTIAD"
KD2_DB01_DB2_RUNLIB         ""

* More rows (one for each subsystem discovered)...

KD2_DB                      END                        * Table end *
```

*Figure 20. RTEDEF(KD5@lpar) member created by the **DISCOVER** action*

The **DISCOVER** action only discovers the parameter values shown in the previous figure in *italics*. You must complete or edit the other parameters.

1 KD2_DB*nn*_DB2_LOADLIB
If an alias is defined for the data set, the alias name will be used for the parameter value instead of the original data set name.

Related tasks

Completing the parameters for discovered Db2 subsystems
The **DISCOVER** action discovers only some of the parameter values required to monitor Db2 subsystems. You must supply the remaining values.

RTEDEF(KI5@lpar)

If the **DISCOVER** action discovers IMS control regions, it creates the RTEDEF(KI5@*lpar*) member. This member contains parameters that configure the IMS monitoring agent.

```
* IMS control regions discovered by KCIOMEGA
* SYSPLEX=sysplex LPAR=lpar DATE=date

KI2_I1                          BEGIN                  * Table begin *

KI2_I101_ROW                    01
KI2_I101_CLASSIC_IMSID          "imsid"
KI2_I101_CLASSIC_MPREFIX        "Mn"
KI2_I101_CLASSIC_GLOBAL         ""
KI2_I101_CLASSIC_STC            "rte_stc_prefixOIn"
KI2_I101_CLASSIC_VTAM_NODE      "rte_vtam_applid_prefixOInN"
KI2_I101_CLASSIC_VTAM_APPL_LOGON "rte_vtam_applid_prefixOIn"
KI2_I101_CLASSIC_IMS_RESLIB     "dsname"  1
KI2_I101_CLASSIC_LROWS          "255"
KI2_I101_CLASSIC_USER_PROFILE   "/C"
KI2_I101_CLASSIC_CTRL_UNIT_ADDR "XXXX"

* More rows (one for each control region discovered)...

KI2_I1                          END                    * Table end *
```

*Figure 21. RTEDEF(KI5@lpar) member created by the **DISCOVER** action*

The **DISCOVER** action only discovers the values of the **KI2_I101_CLASSIC_IMSID** and **KI2_I101_CLASSIC_IMS_RESLIB** parameters.

For other parameters, **DISCOVER** generates placeholder values. Review these values and, if necessary, edit them to match your site requirements.

1 KI2_I1*nn*_CLASSIC_IMS_RESLIB
If an alias is defined for the data set, the alias name will be used for the parameter value instead of the original data set name.

RTEDEF(KMQ#lpar)

If the **DISCOVER** action discovers MQ subsystems, it creates the RTEDEF(KMQ#*lpar*) member. This member contains **SET MANAGER** statements that you can use in the KMQ$CUSR embed override member.

```
* MQ subsystems discovered by KCIOMEGA
* SYSPLEX=sysplex LPAR=lpar DATE=date

SET MANAGER NAME(*)

*SET MANAGER NAME(queue-manager)

* More SET MANAGER statements (one for each queue manager discovered)...
```

*Figure 22. RTEDEF(KMQ#lpar) member created by the **DISCOVER** action*

When discovering MQ subsystems, the **DISCOVER** action creates only a comment member, as indicated by the hash (#) in the member name. This is unlike the discovery of other types of subsystems, where the **DISCOVER** action creates members that contain configuration parameters and have the at sign (@) in the member name. Discovery of MQ subsystems is for informational purposes only.

The IBM OMEGAMON for Messaging on z/OS, IBM MQ Monitoring agent performs its own discovery and, by default, will monitor any queue manager on the LPAR. The default setting of SET MANAGER NAME(*) is recommended because it allows you to change your queue manager configuration without having to change agent parameters.

If you want to name particular queue managers, you can specify multiple statements of SET MANAGER NAME(*queue-manager*).

Because it is a comment member, the information in KMQ#*lpar* is not used for agent configuration directly. However, you can use the information that was discovered and is provided in the KMQ#*lpar* member in your KMQ$CUSR embed override member.

RTEDEF(KN3@lpar)

If the **DISCOVER** action discovers TCP/IP stacks, it creates the RTEDEF(KN3@*lpar*) member. This member contains parameters that configure the networks monitoring agent.

```
* TCP/IP stacks discovered by KCIOMEGA
* SYSPLEX=sysplex LPAR=lpar DATE=date

KN3_TCPX                          BEGIN              * Table begin *

KN3_TCPX01_ROW                    01
KN3_TCPX01_TCP_STC                "task-name"
KN3_TCPX01_SYS_NAME               "lpar"
KN3_TCPX01_TCPIP_PROFILES_DSN     "dsname"  1
KN3_TCPX01_TCPIP_PROFILES_MBR     "member-name"

* More rows (one for each stack discovered)...

KN3_TCPX                          END                * Table end *
```

*Figure 23. RTEDEF(KN3@lpar) member created by the **DISCOVER** action*

1 KN3_TCPX*nn*_TCPIP_PROFILES_DSN
If an alias is defined for the data set, the alias name will be used for the parameter value instead of the original data set name.

RTEDEF(SYS@lpar)

The **DISCOVER** action creates the RTEDEF(SYS@*lpar*) member. This member contains system symbols and KCIPARSE extracted variables needed by the **GENERATE** action.

The primary use is, in conjunction with parameter **KFJ_SYSNAME**, to allow you to pre-generate RTEs on a single local system without having to run the **GENERATE** step on the remote system.

```
* System Symbols discovered by KCIOMEGA
* SYSPLEX=sysplex LPAR=lpar DATE=date

HLQPLEX         "hlqplex"                          'S'
HSMHOST         "2"                                'S'
HSMPRI          "YES"                              'S'
NUMSYS          "2"                                'S'
PLEXNAME        "plexname"                         'S'
RTHLQ           "RSRTE"                            'S'
SMFID           "smfid"                            'S'
SUFFIX          "RS"                               'S'
SYSALVL         "2"                                'S'
SYSCLONE        "B2"                               'S'
SYSLVL          "205"                              'S'
SYSNAME         "name"                             'S'
SYSOSLVL        "Z1020500"                         'S'
SYSPLEX         "sysplex"                          'S'
SYSR1           "RZ205A"                           'S'
UNIXVER         "V2R05"                            'S'
ZDDD            "SAT."                             'S'
...
SYSSMFID        "syssmfid"                         'K'
SYSSMS          "Y"                                'K'
SYSVTAMNETID    "TESTNET1"                         'K'
SYSVTAMSSCP     "RSB2SSCP"                         'K'
SYSIP           "Y"                                'K'
SYSIPADDRESS    "123.456.78"                       'K'
SYSIPADDRESS_F  "123.456.789.012"                  'K'
SYSIPHOSTNAME   "hostname"                         'K'
```

*Figure 24. RTEDEF(SYS@lpar) member created by the **DISCOVER** action*

GENERATE

The **GENERATE** action generates runtime members for a runtime environment from a set of configured parameters.

Before you begin

Before performing a **GENERATE** action for an existing runtime environment, stop the started tasks for that runtime environment. Started tasks can lock runtime members, such as persistent data store data sets. Locked runtime members can cause the **GENERATE** action to fail.

Note: You can also use the **GENERATE** action to make a copy of your SMP/E target libraries. For more information, see "Using SMP/E target library copies" on page 141.

About this task

With the **GENERATE** action, you can generate the runtime members and started tasks for your runtime environment.

The following list provides details about the **GENERATE** action:

- The **GENERATE** action generates runtime members from the parameters in the runtime environment definition library, *rte_plib_hilev*.RTEDEF. The **GENERATE** action builds the set of parameters that is used by concatenating the corresponding RTEDEF library members.
- Run the **GENERATE** action on the LPAR where you will start the runtime environment. For example, if your site uses JES2, insert a **SYSAFF** job parameter after the **JOB** statement to ensure that the job runs on that LPAR.

- If the RTEDEF library contains LPAR-specific members, then the **GENERATE** action uses the LPAR-specific members for the LPAR where the **GENERATE** action is running. For example, consider the RTEDEF library that contains the following members:

  KDS$PARM
  KDS$ZOS1
  KDS$ZOS2

  If you run the **GENERATE** action on LPAR ZOS1, then the **GENERATE** action uses the non-LPAR-specific member KDS$PARM and the LPAR-specific member KDS$ZOS1, but not the LPAR-specific member KDS$ZOS2.
- Specify **KFJ_LOCAL_PLIB_HILEV** in the KCIVARS DD, along with **KFJ_SYSNAME**, if you want to generate a runtime environment that will be deployed using a different local high-level qualifier.

  If you decide to use different settings for the local generation of the runtime environment, there are certain limitations in terms of parameters that can be customized. The following parameters are not allowed in the respective RTEDEF members:

  KYN_XAI0_SUBAGENT_PRODHOME
  KQI_HFS_HFSROOT_DIR1
  KS3_APP_ZFS_DIR
  KM2_HIST_DSTOR_RKM2EDS_DSN*x* (where *x* is 1 to 7)
  GBL_USER_JCL
  KD2_OMPE_DSHLQ
  KD2_OMPE_VSAM_DSHLQ

  If any of these parameters are explicitly specified in the RTEDEF members, message KFJ00213E is returned in KCIPRINT and the workflow stops. To continue, remove these parameters from the RTEDEF data set members and re-run the **GENERATE** action.
- You can use the **OPTION** parameter for the **GENERATE** action to control certain processing. For information about the available options, see "GENERATE options" on page 47.

To generate runtime members for a runtime environment using the **GENERATE** action, use the following procedure.

Procedure

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a **GENERATE** action.
2. Specify values for the required parameters **RTE_NAME** and **RTE_PLIB_HILEV**.
3. (Optional) To generate a runtime environment that will be deployed using a different local high-level qualifier, specify **KFJ_LOCAL_PLIB_HILEV** and **KFJ_SYSNAME**.
4. (Optional) To validate your RTEDEF data set before running the **GENERATE** action, add **OPTION VALIDATE**, then run the KFJJMCM job and review the validation report defined in the $VALRPT DD statement.

   Note: You cannot use the **VALIDATE** option with any other **OPTION** value.
5. (Optional) To bypass configuration processing for z/OS UNIX System Services, specify **OPTION NOUSS**.
6. To run the **GENERATE** action to create the runtime members, remove **OPTION VALIDATE** (if present), and then run the KFJJMCM job

   Job messages for the **GENERATE** action are written to the KCIPRINT SYSOUT data set.
7. (Optional) On subsequent runs of the **GENERATE** action, you can perform select configuration steps by specifying the following options, as needed. To specify multiple options, separate the values with a comma and no spaces.
   - To run only the **GENERATE** workflow stage that deploys the parts related to z/OS UNIX, specify **OPTION USS**.
   - To perform configuration for security exits only, specify **OPTION SECEXITS**.

- To perform only the step that loads the read-only configuration members to the RK* data sets, specify **OPTION QUICKLOAD**.

  Note: You can use the **QUICKLOAD** option with the **USS** and **SECEXITS** options, but you cannot use it with the **NOUSS** option.
- To update the configurable members for the runtime environment (for example, in the RKANPARU, RKANSAMU, and RKANCMDU libraries) without refreshing data from SMP/E target libraries, specify **OPTION QUICKCONFIG**.

  Note: You cannot use the **QUICKCONFIG** option with the **VALIDATE** option or the **NOUSS** option.
- To only assemble and link elements, specify **OPTION RELINK**.

Example

The following JCL generates runtime members for the runtime environment that is defined by members of the TSOUID.MONSUITE.RTEDEF library, including RTE1 and LPAR-specific configuration profile members such as K*pp*$ZOS1.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            GENERATE
RTE_NAME          RTE1
RTE_PLIB_HILEV    TSOUID.MONSUITE
/*
```

*Figure 25. Example JCL to perform the **GENERATE** action*

Note the JES2 **SYSAFF** parameter that causes the job to run on LPAR ZOS1.

Related reference

Runtime members
The **GENERATE** action generates runtime members in locations (MVS data set names and z/OS UNIX System Services paths) that are specified by parameters.

Runtime environment definition library members
RTEDEF library members follow a naming convention that identifies the contents of the member and whether the member applies to a specific LPAR or to all LPARs.

GENERATE options

You can use the **OPTION** parameter for the **GENERATE** action to control certain processing. This topic describes the available options for use with the **GENERATE** action.

Overview of GENERATE options

Table 6 on page 47 provides an overview of the **GENERATE** options that are available. Each of the available options is described in more detail in "GENERATE OPTION keywords" on page 48.

Table 6. Overview of GENERATE options

OPTION	Abbreviation	Description
USS	–	Run only the **GENERATE** workflow stage that deploys the parts related to z/OS UNIX System Services. This option requires a complete run of the **GENERATE** action prior to use.
NOUSS	–	Do not run the z/OS UNIX deploy stage in the **GENERATE** action.
SECEXITS	SEC	Perform configuration for security exits only. This option requires a complete run of the **GENERATE** action prior to use.

Table 6. Overview of GENERATE options (continued)		
OPTION	**Abbreviation**	**Description**
VALIDATE	VAL	Perform initial validation of RTEDEF parameters.
QUICKLOAD	QL	Load the read-only configuration members to the RK* data sets. The read-only members are those members that are not impacted by customization during configuration. This option requires a complete run of the **GENERATE** action prior to use.
QUICKCONFIG	QC	Update the configurable members for the runtime environment (for example, in the RKANPARU, RKANSAMU, and RKANCMDU libraries) without refreshing data from SMP/E target libraries. This option requires a complete run of the **GENERATE** action prior to use.
RELINK	LINK	Assemble and link edit modules for OMEGAMON for Networks (KN3) and OMEGAMON enhanced 3270 user interface (KOB). This option requires a complete run of the **GENERATE** action prior to use.
TRGCOPY	TRG	Make a copy of your SMP/E target libraries, from which you can create or update your runtime environments.

To specify more than one option, separate the values with a comma and no spaces. For example: **OPTION USS,SECEXITS**

Compatibility of GENERATE options

Some options are not compatible to run with other options during the same job. Table 7 on page 48 indicates which options are compatible. **Y** in the table cell indicates that you can specify the two options together in the same job. **N** in the table cell indicates that you cannot specify the two options together in the same job.

Table 7. Compatibility of GENERATE options								
	USS	**NOUSS**	**SECEXITS**	**VALIDATE**	**QUICKLOAD**	**QUICKCONFIG**	**RELINK**	**TRGCOPY**
USS	—	N	Y	N	Y	Y	Y	N
NOUSS	N	—	N	N	N	N	N	Y
SECEXITS	Y	N	—	N	Y	Y	Y	N
VALIDATE	N	N	N	—	N	N	N	N
QUICKLOAD	Y	N	Y	N	—	Y	Y	N
QUICKCONFIG	Y	N	Y	N	Y	—	Y	N
RELINK	Y	N	Y	N	Y	Y	—	N
TRGCOPY	N	Y	N	N	N	N	N	—

GENERATE OPTION keywords

USS | NOUSS

You can optionally control when to run the **GENERATE** workflow stage that deploys the parts related to z/OS UNIX. By default, the **GENERATE** action automatically performs any required z/OS UNIX configuration. However, there might be cases where you would want to skip this step (for reasons such as authorization issues), and perform this step at a later time. You can use the **OPTION** parameter to specify the **NOUSS** or **USS** value to control this processing, as follows:

NOUSS
When **OPTION NOUSS** is specified, the **GENERATE** action does not run the stage that deploys the parts related to z/OS UNIX. You can use this option on the initial run of the **GENERATE** action for a new runtime environment, as well an on subsequent runs.

USS
When **OPTION USS** is specified, the **GENERATE** action runs only the stage that deploys the parts related to z/OS UNIX; no other processing is performed (unless another **OPTION** value is also specified, such as **SECEXITS**). You cannot use this option on the initial run of the **GENERATE** action for a new runtime environment; it can only be used on subsequent runs.

Important:

- You must perform a complete run of the **GENERATE** action before you can use **OPTION USS**.
- If you have changed any parameters inside your RTEDEF members that impact z/OS UNIX configuration, do not use **OPTION USS**, because it will not capture your parameter changes. Instead, run action **GENERATE** with **OPTION NOUSS** (or without the option) to reconfigure the members related to z/OS UNIX. Then, you can use **OPTION USS** to move the updated members to z/OS UNIX.

SECEXITS

You can perform configuration processing for security exits separately from the rest of the of **GENERATE** workflow. By default, the **GENERATE** action automatically performs the required configuration tasks for security exits, which includes rebuilding and relinking the security exits. By performing these tasks separately, you can save valuable CPU cycles.

When **OPTION SECEXITS** is specified, the **GENERATE** action performs configuration for security exits only; no other processing is performed (unless another **OPTION** value is also specified, such as **USS**). You cannot use this option on the initial run of the **GENERATE** action for a new runtime environment; it can only be used on subsequent runs.

Important: You must perform a complete run of the **GENERATE** action before you can use the **SECEXITS** option.

For more information about creating your security exits library, see "Setting up security exits in your runtime environment" on page 131.

VALIDATE

You can verify that all parameters specified in the RTEDEF data set have correct values and that you are not missing anything. This option helps you to prepare for running the full **GENERATE** action. **OPTION VALIDATE** will re-create interim and work libraries and will perform validation steps to ensure that your configuration is correct. You cannot use the **VALIDATE** option with any other **OPTION** value. After running the **GENERATE** action with **OPTION VALIDATE**, you can find the validation report using the $VALRPT DD statement.

Attention: If you previously used PARMGEN for configuration, be aware that using the **GENERATE** action with **OPTION VALIDATE** will re-create the WCONFIG data set, which will invalidate the PARMGEN configuration. This means that if you migrate a runtime environment from PARMGEN to Configuration Manager using the **MIGRATE** action, and then perform the **GENERATE** action with **OPTION VALIDATE**, you can no longer use the PARMGEN configuration.

QUICKLOAD

Using **OPTION QUICKLOAD**, you can load the read-only configuration members to the RK* data sets. The read-only members are those members that are not impacted by customization during configuration.

OPTION QUICKLOAD is compatible with the **USS** and **SECEXITS** options, but is not compatible with the **VALIDATE** and **NOUSS** options. This option requires a complete run of the **GENERATE** action prior to use.

QUICKCONFIG

With this option, you can quickly reconfigure your products without reloading the read-only runtime members or refreshing data from SMP/E target libraries. Using this option can improve performance, with the most impact experienced when using a full runtime environment (**RTE_TYPE** parameter is set to FULL).

When **OPTION QUICKCONFIG** is specified, the **GENERATE** action updates the configurable members for the runtime environment (for example, in the RKANPARU, RKANSAMU, and RKANCMDU libraries) without refreshing data from SMP/E target libraries. The read-only members, which are members that are not impacted by customization during configuration, are not loaded to the RK* data sets.

More specifically, running the **GENERATE** action with **OPTION QUICKCONFIG** performs the following actions:

- Validates parameters
- Allocates missing data sets
- Updates configurable members for the runtime environment (for example, in the RKANPARU, RKANSAMU, and RKANCMDU libraries)
- Updates started tasks, VTAM list, and VTAM node
- Updates z/OS UNIX-related members, but does not deploy them to the z/OS UNIX directories or files. To deploy prepared members to z/OS UNIX files or directories, you must combine the **QUICKCONFIG** option with the **USS** option.

Running the **GENERATE** action with **OPTION QUICKCONFIG** does not perform the following actions:

- Refresh data from SMP/E target libraries
- Update read-only modules from SMP/E target libraries
- Update z/OS UNIX files or directories
- Perform any assembly or linking operations related to security exits or other load modules

You cannot use the **QUICKCONFIG** option with the **VALIDATE** option or the **NOUSS** option. The **QUICKCONFIG** option requires a complete run of the **GENERATE** action prior to use.

RELINK

Use **OPTION RELINK** to assemble and link elements into the SYSLMOD RKANMOD* load library when a relink of IBM Z OMEGAMON AI for Networks (KN3) or OMEGAMON Enhanced 3270 User Interface (KOB) modules is required, such as when applying maintenance.

Note: If you are familiar with PARMGEN, the **RELINK** option performs the same function as the PARMGEN KCIJPLNK job (**Run post-SMP/E RKANMODU ASM/LINK steps**).

Tip: You can customize the binder program that is used by specifying the program name in parameter "GBL_UTIL_BINDER" on page 81.

When using the **GENERATE** action with **OPTION RELINK**, the details differ depending on the environment where the action runs, as follows:

On a local system (the *configuration system* in a remote deployment scenario)

You can run the **GENERATE** action with **OPTION RELINK** on a local system to assemble and link elements.

On a remote environment with a different z/OS level

You can run the **GENERATE** action with **OPTION RELINK** on a target system that is at a different z/OS level than the configuration system. You do not need to run all the steps in the remote deployment scenario (**GENERATE** action, **PACKAGE** action, transfer data sets, **DEPLOY** action).

Using the **RELINK** option requires the SMP/E library TKANMODS to be present on the target system. The TKANMODS library is not part of the full runtime environment, and it is not created by the **BLDREMDS** action. In other words, you must have an SMP/E environment present to ensure consistency. As a result, you must run Configuration Manager from the same SMP/E libraries.

You can rerun the **GENERATE** action with **OPTION RELINK** as needed.

When you use the **RELINK** option to relink modules on a remote environment at a different z/OS level, you might have different settings in the libraries set by parameters **GBL_DSN_SYS1_MACLIB** and **GBL_DSN_CEE_SCEELKED** than what you would have on your local system. If there are significant differences in the contents of those libraries, which control how modules are being assembled and linked, then you should run the **GENERATE** action with **OPTION RELINK** on the target system.

On a remote environment with the same z/OS level

You can run the **GENERATE** action with **OPTION RELINK** on a target system that is at the same z/OS level as the configuration system. In this environment, there is not an SMP/E environment on the target system, and only the minimal set of libraries is transferred to perform the **DEPLOY** action.

You can run the following series of steps:

1. **GENERATE** action with **OPTION RELINK**
2. **PACKAGE** action
3. Transfer of data sets
4. **DEPLOY** action

This implementation saves CPU cycles on the **GENERATE** step.

For a full runtime environment

In a full runtime environment, you must run the **GENERATE** action with **OPTION QUICKLOAD,RELINK** to achieve the correct relinking results. (This requirement is due to a dependency on the RKANSAM data set).

TRGCOPY

Use **OPTION TRGCOPY** to make a copy of your SMP/E target libraries, from which you can create or update your runtime environments. You can abbreviate this keyword to **TRG**. For more information, see "Using SMP/E target library copies" on page 141.

DELETE

Use the **DELETE** action to delete the runtime libraries for your runtime environment.

Before you begin

Review the following information before you use the **DELETE** action:

- The **DELETE** action requires the **RTE_NAME** and **RTE_PLIB_HILEV** parameters and values.
- If you are implementing a remote deployment scenario and used the **KFJ_LOCAL_PLIB_HILEV** parameter when creating the runtime environment, you can also specify this parameter for the **DELETE** action, depending on your situation. For more information, see "Deleting libraries used for remote deployment" on page 53.
- Review details about the data sets that are deleted by the **DELETE** action and those that are not affected, as follows:

  Data sets that are deleted by the DELETE action

  The **DELETE** action deletes the runtime libraries created by the **GENERATE** action. This includes the PARMGEN libraries that the **GENERATE** action creates: WCONFIG, interim staging, work, and global user JCL.

  The following list describes the name patterns of the data sets that are affected by the **DELETE** action:

  – Data sets that match the following name patterns:

    *rte_plib_hilev.rte_name.**

rte_hilev.*rte_name*.*
rte_vsam_hilev.*rte_name*.*

By default, these are all the same pattern, because the default value for *rte_hilev* and *rte_vsam_hilev* is *rte_plib_hilev*.

- For the following data sets, the default values for these parameters, when not specified explicitly, include *rte_name*, but custom values are allowed:

  rte_pds_hilev.*
  rte_pds2_hilev.*

- Data sets that match the following product-specific name patterns, if the respective products have been configured:

  IBM OMEGAMON for Db2 Performance Expert on z/OS

  KD2_OMPE_DSHLQ.*
  KD2_OMPE_VSAM_DSHLQ.*

  IBM OMEGAMON for IMS on z/OS

  KI2_LOGR_EHLQ.KI2_LOGR_LS_PREFIX.*

- For remote deployment scenarios, if you used the **KFJ_LOCAL_PLIB_HILEV** parameter for deploying remote environments, data sets that match the following name patterns:

  kfj_local_plib_hilev.*rte_name*.*
  kfj_local_hilev.*rte_name*.*
  kfj_local_vsam_hilev.*rte_name*.*
  kfj_local_pds_hilev.*

  Note: When deploying remote environments, the *rte_** parameters are for the remote runtime environments and the *kfj_local_** parameters are for the local runtime environments. On a single run of the **DELETE** action, either *rte_** parameter values or *kfj_local_** parameter values are used for the high-level qualifier.

Data sets not affected by the DELETE action

The **DELETE** action does not affect the following data sets:

- The runtime environment definition library: *rte_plib_hilev*.RTEDEF
- The security exits library, defined by **RTE_X_SECURITY_EXIT_LIB**
- The embed overrides library, defined by **RTE_X_OVERRIDE_EMBEDS_LIB**
- Any other data sets, such as persistent data store data sets, that have been allocated outside the *rte_name*-based data set name patterns described previously
- z/OS UNIX System Services directories

About this task

The following list provides details about the **DELETE** action:

- The **DELETE** action uses the specified information and values in the configuration members to derive the names of the data sets to be deleted.
- You must perform the **DELETE** action on the system where the files to be deleted exist, which is significant in remote deployment scenarios.
- The **DELETE** action requires the **CONFIRM** workflow variable to delete the data sets. You can also use the **CONFIRM** parameter to preview the data sets that will be deleted. Specify the **CONFIRM** parameter in the KCIVARS DD statement with one of the following values:

  N
  (Default) List all data sets that will be deleted. The list appears in KCIPRINT.

Important: It is recommended that you review the list of data sets that will be deleted before performing the delete.

Y

Delete the data sets. The **DELETE** action deletes your runtime libraries, as described in "Data sets that are deleted by the DELETE action" on page 51.

Procedure

Perform the following steps on the system where the files to be deleted exist:

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a **DELETE** action.
2. Add the **CONFIRM** workflow variable and set to N, which will allow you to review the list of data sets that will be deleted.
3. Specify values for the required parameters **RTE_NAME** and **RTE_PLIB_HILEV**.
4. For remote deployment scenarios: If you are using remote deployment and want to delete data sets on the configuration LPAR, specify the **KFJ_LOCAL_PLIB_HILEV** parameter and value.
5. Run the KFJJMCM job to display the data sets that will be deleted.

   The list appears in the KCIPRINT SYSOUT data set.
6. Review the generated list of data sets that will be deleted.
7. If you are satisfied with the list of data sets that will be deleted, change the **CONFIRM** workflow variable to Y and run the KFJJMCM job.

   The **DELETE** action deletes your runtime libraries.

Example

The following JCL deletes the runtime libraries for the runtime environment that is defined in the member TSOUID.MONSUITE.RTEDEF(RTE1).

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            DELETE
CONFIRM           Y
RTE_NAME          RTE1
RTE_PLIB_HILEV    TSOUID.MONSUITE
/*
```

*Figure 26. Example JCL to perform the **DELETE** action*

Note the JES2 **SYSAFF** that causes the job to run on LPAR ZOS1. This is required only if the **RTE_NAME** parameter in the TSOUID.MONSUITE.RTEDEF(RTE1) member refers to a variable whose value is LPAR-specific.

Deleting libraries used for remote deployment

You can use the **DELETE** action to delete libraries used for remote deployment.

The process of deploying a remote environment using the Configuration Manager **PACKAGE** and **DEPLOY** actions results in multiple sets of data sets on multiple systems. In a remote deployment scenario, you create a runtime environment on a specific LPAR (the configuration LPAR), package the runtime environment data sets into transferable dump data sets on the configuration LPAR, transfer the packaged data sets to the remote target system (target LPAR), and deploy the packaged runtime environment data sets on the target LPAR. As mentioned previously, many data sets are created; and these data sets can also have different high-level qualifiers. Deleting these data sets manually would be a complex task. Instead, you can delete these data sets easily and accurately using the **DELETE** action.

Using the **DELETE** action, you can remove these data sets on the configuration LPAR or the target LPAR, and you must perform the **DELETE** action on the system where the files to be deleted exist. If you used

the **KFJ_LOCAL_PLIB_HILEV** parameter when creating the runtime environment, the **DELETE** action identifies and deletes the runtime environment that was generated for a remote system using different high-level qualifiers.

Important: It is recommended that you review the list of data sets that will be deleted by using the **CONFIRM** workflow variable set to N before performing the delete. After you are satisfied with the generated list, run the **DELETE** action using the **CONFIRM** workflow variable set to Y to delete the data sets.

The following examples describe which parameters to use with the **DELETE** action depending on your situation and assumes that you used **KFJ_LOCAL_PLIB_HILEV** when creating your runtime environment.

Note: For information about the data sets that are affected, see "Data sets that are deleted by the DELETE action" on page 51.

Deleting data sets on the configuration LPAR

To delete runtime environment data sets on the configuration LPAR, run the **DELETE** action on the configuration LPAR using the parameters in the following example. To indicate the high-level qualifier, note that both the **RTE_PLIB_HILEV** and **KFJ_LOCAL_PLIB_HILEV** parameters are specified.

```
//KCIVARS  DD *
ACTION                  DELETE
CONFIRM                 Y
RTE_NAME                RTE1
RTE_PLIB_HILEV          TSOUID.TARG1
KFJ_LOCAL_PLIB_HILEV    TSOUID.LOCL1
```

Figure 27. Example to delete data sets on the configuration LPAR

This example will use the `TSOUID.LOCL1.RTEDEF(PCK$*)` members to generate the list of data sets to delete on the configuration LPAR.

Important: Make sure to specify the correct values for the **RTE_PLIB_HILEV** and **KFJ_LOCAL_PLIB_HILEV** parameters; otherwise, unexpected results might occur. It is important that you review the generated list of data sets to be deleted using the **CONFIRM** workflow variable set to N.

Deleting data sets on the remote system (target LPAR)

To delete runtime environment data sets on the target LPAR, run the **DELETE** action on the target LPAR using the parameters in the following example. To indicate the high-level qualifier, note that only the **RTE_PLIB_HILEV** parameter is specified.

```
//KCIVARS  DD *
ACTION                  DELETE
CONFIRM                 Y
RTE_NAME                RTE1
RTE_PLIB_HILEV          TSOUID.TARG1
```

Figure 28. Example to delete data sets on the remote system (target LPAR)

This example will delete the data sets for the runtime environment on the target LPAR with high-level qualifier `TSOUID.TARG1`, or as defined in `TSOUID.TARG1.RTEDEF(RTE1)`.

Important: Make sure to specify the correct value for the **RTE_PLIB_HILEV** parameter. If you inadvertently specify the value that was used for parameter **KFJ_LOCAL_PLIB_HILEV** for parameter **RTE_PLIB_HILEV**, unexpected results might occur. It is important that you review the generated list of data sets that will be deleted using the **CONFIRM** workflow variable set to N.

MIGRATE

The **MIGRATE** action imports configuration settings from a runtime environment that is configured with PARMGEN to one that is configured with Configuration Manager.

Before you begin

Review the following information before you use the **MIGRATE** action:

- If you migrate a runtime environment that is configured with PARMGEN to one that is configured with Configuration Manager, you can no longer use PARMGEN to configure the runtime environment. For more information, see "Comparison with PARMGEN" on page 5.
- In this task, *source* refers to the runtime environment that is configured with PARMGEN, and *target* refers to the runtime environment that is configured with Configuration Manager.
- Migration works only for OMEGAMON products that are supported by Configuration Manager. If the migration source contains other products configured by PARMGEN that are not supported by Configuration Manager, error message KFJ00001E is issued for the **MIGRATE** action in the KCIPRINT output, and the job ends. For the list of supported products, see "Products supported by Configuration Manager" on page 1.
- Consider the naming convention that you will use for your target runtime environment. Source and target runtime environments can share the same high-level qualifier, which is referred to as an *in-place migrate*.

  Note: Whereas PARMGEN stores parameters in *rte_plib_hilev*.*rte_name*.WCONFIG for each runtime environment, Configuration Manager stores parameters and variables in *rte_plib_hilev*.RTEDEF, which can contain definitions for multiple runtime environments. For more information about the differences between PARMGEN and Configuration Manager, see "Comparison with PARMGEN" on page 5.
- The **MIGRATE** action supports migrating one or multiple PARMGEN runtime environments into a single Configuration Manager RTEDEF configuration. It is recommended that you decide prior to the migration of your first runtime environment whether you plan to migrate one or multiple runtime environments into a single RTEDEF.

  Note: If you are going to set up a High Availability TEMS (HA TEMS), make sure only one runtime environment is defined in the RTEDEF (that is, the one used for the HA TEMS).
- For parameters that describe data set qualifiers, the **MIGRATE** action does not migrate parameters that have been customized with hardcoded values that partially match the PARMGEN **RTE_HILEV** parameter value. For example, if in your PARMGEN configuration, you have set parameter **KD2_OMPE_DSHLQ** to use value TEST.RTE1.HLQ1, and **RTE_HILEV** is set to "TEST.RTE1", then **KD2_OMPE_DSHLQ** will not be migrated.

  Before migrating from PARMGEN to Configuration Manager, review your customized parameters that describe data set qualifiers. If you have customized data set names that partially match your PARMGEN configuration **RTE_HILEV** value, but use a hardcoded value instead of parameter reference **%RTE_HILEV%**, you must update the value to use **%RTE_HILEV%** in order for the parameter to migrate successfully.
- Before performing a **MIGRATE** action, make sure you have a backup of your source PARMGEN runtime environment. The next step in Configuration Manager after a migration is to generate runtime members using the **GENERATE** action. If you perform an in-place migration, the subsequent **GENERATE** action will overwrite the runtime environment data sets that were used by PARMGEN.

About this task

With the **MIGRATE** action, you can import existing PARMGEN runtime environment configuration settings from a specific WCONFIG member into the Configuration Manager *rte_plib_hilev*.RTEDEF. The **MIGRATE** action reads the WCONFIG and other data sets from a PARMGEN installation, from which it creates the sparse descriptors containing the parameters, hiding every parameter setting that is

considered a default or has not been changed. It also copies other files for system variables support, embed overrides, and security exits that are required to support the migration.

The following list provides details about the **MIGRATE** action:

- The **MIGRATE** action supports migrating one PARMGEN runtime environment at a time.
- You can migrate one or more PARMGEN runtime environments into a single Configuration Manager RTEDEF configuration. The default behavior of the **MIGRATE** action is to migrate only one runtime environment into the RTEDEF data set. Using the **OPTION MULTIPLE** parameter, you can migrate multiple runtime environments into a single RTEDEF data set. Each runtime environment migration requires a separate **MIGRATE** action job. If you plan to migrate multiple runtime environments into a single RTEDEF data set, make sure to include the **OPTION MULTIPLE** parameter on every **MIGRATE** action job, including the first one.

  Note: You can abbreviate **OPTION MULTIPLE** to **OPTION MULTI**.
- The **MIGRATE** action creates the necessary members in the RTEDEF data set, as follows:
  - When using the default behavior of the **MIGRATE** action to migrate one runtime environment into a single RTEDEF data set (omitting the **OPTION MULTIPLE** parameter), the **MIGRATE** action will create members of type K*pp*$PARM in the respective created RTEDEF data set, along with the *rte_name* member for the runtime environment-specific parameters.
  - When migrating multiple runtime environments into a single RTEDEF configuration, use parameter **OPTION MULTIPLE** and **KFJ_SYSNAME lpar** in KCIVARS DD. The **MIGRATE** action will create members of type K*pp*$*lpar* in the RTEDEF data set, along with the *rte_name* member for the runtime environment-specific parameters.

    On subsequent runs of the **MIGRATE** action, reuse the same **RTE_PLIB_HILEV** parameter value, but update the values for parameters **RTE_NAME**, **KFJ_MIGRATE_WCONFIG**, and **KFJ_SYSNAME** to create a new set of runtime environment parameter members. There is no limit on how many runtime environments can be migrated into a single RTEDEF data set.
- If the **MIGRATE** action detects that a specified target RTEDEF already contains K*pp*$*lpar* and *rte_name* members, **MIGRATE** issues an error message and stops. Note that **MIGRATE** will detect K*pp*$PARM and VAR$GLOB members in RTEDEF as well. Because these members are considered to have a sysplex scope, they can only exist in RTEDEF during the migration process if no additional K*pp*$*lpar* members are intended to be migrated into the same RTEDEF. Depending on the case, any of the following messages might appear: KFJ00218E, KFJ00219E, KFJ00220E
- The **MIGRATE** action accepts PARMGEN runtime environments with system variables. However, system variables are not copied unless you have chosen to override them in your PARMGEN configuration. Variables are copied to the RTEDEF member VAR$GLOB for a default (single) **MIGRATE** action or member VAR$*lpar* in a multiple **MIGRATE** action.
- The **MIGRATE** action allocates the security exits library with the default name *rte_plib_hilev*.*rte_name*.SECEXITS (or, optionally, the name specified in the **KFJ_SECURITY_EXITS_LIB** parameter). The **MIGRATE** action also copies the security exits used by the PARMGEN environment to the specified security exits library, and defines the source security exits library to the runtime environment using the **RTE_X_SECURITY_EXIT_LIB** parameter. For more information, see "Setting up security exits in your runtime environment" on page 131.

  Important: The **RTE_X_SECURITY_EXIT_LIB** parameter will contain the name of the security exits library used by the source PARMGEN environment; you must review this setting and update it if necessary before running the **GENERATE** action.
- If the use of override embed members is enabled by specifying parameter **KFJ_USE_EMBEDS** set to Y, the **MIGRATE** action allocates the embeds data set with the default name *rte_plib_hilev*.*rte_name*.EMBEDS (or, optionally, the name specified in the **KFJ_EMBEDS_LIB** parameter). The **MIGRATE** action sets up the embeds data set, populates it with supported override embed parameters (if applicable), and defines it to the runtime environment using the **RTE_X_OVERRIDE_EMBEDS_LIB** parameter. For more information, see "Using override embed members" on page 133.

- The **MIGRATE** action works with the **KFJ_LOCAL_PLIB_HILEV** parameter to allow for local generation of runtime environments for remote systems using different high-level qualifiers.

  When the **KFJ_LOCAL_PLIB_HILEV** parameter is specified, the generated *kfj_local_plib_hilev*.RTEDEF data set will contain an additional member: PCK$PARM for a default (single) **MIGRATE** action, or member PCK$*lpar* in a multiple **MIGRATE** action. This member allows locally generated runtime environments using a different data set high-level qualifier than the one intended to be used on the deployment target (for example, the production system).

  For more information about remote deployments, see "Special considerations for SYSPLEX rollout" on page 111, "RTEDEF(PCK$PARM)" on page 114, and "Remote deployment scenario" on page 137.

After you run the **MIGRATE** action, you must carefully review the generated RTEDEF data set members to verify that the parameters have the expected values. You can use the report provided in the **MIGRATE** job output identified by the MIGRPT DD statement to review details about the parameters. This report presents parameters in the following groups:

- Parameters that are migrated to the RTEDEF data set because their values are different from the default values
- Parameters that are always migrated, regardless of values being default or not
- Parameters that are not migrated because of having default values
- Parameters that are not migrated because they match the PARMGEN **RTE_HILEV** parameter value

This report provides each parameter with its resolved value. Note that some parameters use system variables or are dependent on other parameters, and some parameters inherit values from other parameters and might be regarded as having default values. Additionally, some parameters have different default values than PARMGEN, as outlined in "Parameters with different default values than PARMGEN" on page 80.

Important: Verify that the data set high-level qualifiers, data set names, and z/OS UNIX System Services paths are correct, as a subsequent **GENERATE** action might overwrite existing files.

To migrate a PARMGEN runtime environment to a Configuration Manager runtime environment, use the following procedure.

Note: The **CONFIRM** workflow variable is not supported for the **MIGRATE** action.

Procedure

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a **MIGRATE** action.
2. Specify the required parameter values, as follows:
   a) Specify values for the target environment that is to be configured with Configuration Manager in parameters **RTE_NAME** and **RTE_PLIB_HILEV**.
   b) Specify the name of the source WCONFIG data set in the **KFJ_MIGRATE_WCONFIG** parameter. This is the WCONFIG data set of the PARMGEN-configured runtime environment from which configuration settings are to be imported.
3. (Optional) Specify additional parameters as needed, for example:
   - To migrate this runtime environment into a RTEDEF data set containing multiple runtime environment configurations, add the **OPTION MULTIPLE** and **KFJ_SYSNAME lpar** parameters. If this is a subsequent run of the **MIGRATE** action, reuse the same **RTE_PLIB_HILEV** parameter value, but update the values for parameters **RTE_NAME**, **KFJ_MIGRATE_WCONFIG**, and **KFJ_SYSNAME**.
   - To specify a different name for the security exits library, add the **KFJ_SECURITY_EXITS_LIB** parameter and value.
   - To enable the use of override embed members, add the **KFJ_USE_EMBEDS** parameter set to Y and the **KFJ_EMBEDS_LIB** parameter and value.
4. Run the KFJJMCM job to perform the migration and generate the new RTEDEF data set.
   Job messages for the **MIGRATE** action are written to the KCIPRINT SYSOUT data set.

5. Review the generated RTEDEF data set members to verify that the parameters have the expected values. Verify that the data set high-level qualifiers, data set names, and z/OS UNIX paths are correct. You can also use the report provided in the **MIGRATE** job output identified by the MIGRPT DD to review details about the parameters.

Example

The following JCL jobs migrate an existing PARMGEN configuration pointed to by *highlevel*.WCONFIG into an RTEDEF library TSOUID.MONSUITE.RTEDEF. The first example is for a single runtime environment RTEDEF, and the second example is for a multiple runtime environment RTEDEF. These examples also specify that override embed members are enabled and provide custom data set names for the security exits and embeds libraries.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
*
ACTION                  MIGRATE
RTE_NAME                RTE1
RTE_PLIB_HILEV          TSOUID.MONSUITE

KFJ_MIGRATE_WCONFIG     highlevel.WCONFIG

KFJ_SECURITY_EXITS_LIB  TEST1.TST.DEMO.MYEXITS
KFJ_USE_EMBEDS          Y
KFJ_EMBEDS_LIB          TEST1.TST.DEMO.MYEMBEDS
/*
```

*Figure 29. Example JCL to perform the **MIGRATE** action for a single runtime environment RTEDEF*

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
*
ACTION                  MIGRATE
OPTION                  MULTIPLE
RTE_NAME                RTE1
RTE_PLIB_HILEV          TSOUID.MONSUITE

KFJ_MIGRATE_WCONFIG     highlevel.WCONFIG

KFJ_SECURITY_EXITS_LIB  TEST1.TST.DEMO.MYEXITS
KFJ_USE_EMBEDS          Y
KFJ_EMBEDS_LIB          TEST1.TST.DEMO.MYEMBEDS

KFJ_SYSNAME             &SYSNAME
/*
```

*Figure 30. Example JCL to perform the **MIGRATE** action for a multiple runtime environment RTEDEF*

What to do next

After you have finished migrating your PARMGEN runtime environments into the Configuration Manager RTEDEF library, use the **GENERATE** action to generate runtime members using the configured parameters. See GENERATE.

Related reference

Using override embed members

With Monitoring Configuration Manager, you can use override embed members to provide and maintain customization for your runtime environments.

PACKAGE

The **PACKAGE** action packages a runtime environment that can then be deployed to a remote system.

Before you begin

The **PACKAGE** action can be run after you have successfully generated a runtime environment using the **GENERATE** action.

Important: Before you run the **PACKAGE** action, perform the following important steps:

- For an existing runtime environment, stop the started tasks for that runtime environment. Started tasks can lock runtime members, which can cause the **PACKAGE** action to fail.
- For the SMS-managed data sets for the runtime environment to be packaged, make sure that any migrated data sets are recalled.

For more information about the complete remote deployment process, see "Remote deployment scenario" on page 137.

About this task

The **PACKAGE** action will use the DFSMSdss DUMP ADRDSSU program to create a series of data sets that contain the runtime environment. These dump files can be transferred across SYSPLEX boundaries, using methods like FTPS. The **PACKAGE** action is used in conjunction with the **DEPLOY** action.

The following list provides details about the **PACKAGE** action:

- The **PACKAGE** action allocates the package files using the following default names:

  ```
  RTE_PLIB_HILEV.RTE_NAME.PACK<xx>.DMP
  ```

  Where <*xx*> is one of the following package codes:

  MN
  Main non-VSAM package. Contains data sets from **RTE_HILEV** and some from **RTE_PLIB_HILEV**.

  MV
  Main VSAM package. Contains non-history related VSAM data sets from **RTE_VSAM_HILEV**.

  HN
  History non-VSAM package. Contains history-related data sets from **RTE_PDS_HILEV**.

  HV
  History VSAM package. Contains history-related data sets from **RTE_VSAM_HILEV**.

  The **PACKAGE** action also creates a metadata file named RTE_PLIB_HILEV.RTE_NAME.PACKMD. This is a flat file that is not tersed or dumped. It is required for the **DEPLOY** action if a data set is renamed.
- The **PACKAGE** action requires the use of the KCIALPHA program. KCIALPHA is an APF-authorized version of KCIOMEGA.
- You can specify the following optional parameters in the KCIVARS DD statement when running the **PACKAGE** action:

  KFJ_PACK_HILEV
  Overrides package HILEV to KFJ_PACK_HILEV.RTE_NAME.PACKxx.*. To avoid exceeding the z/OS 44-character limit for data set names, the combined length of **RTE_NAME** and **KFJ_PACK_HILEV** should not exceed 28 characters. For example, if **RTE_NAME** is 8 characters, then **KFJ_PACK_HILEV** should not exceed 20 characters.

  KFJ_ADRDSSU_ADMIN
  Specifies if the ADMINISTRATOR option should be used with EMBEDS ADRDSSU. Use Y/N values. The default is N. This option may be required in some sites depending on their security settings.

KFJ_PACK_TERSE
Specifies if the DMP data sets should also be tersed. Use Y/N values. The default is N. Specify KFJ_PACK_TERSE Y if you want to transfer the packages using FTP. In this case, the DMP files will be tersed and only the tersed copy will be retained.

Note: If you are using virtual tapes to transfer your RTE data sets, tersing the dump files is not necessary.

If specified, the following parameters will be used for both the DMP and DMP.TRS data sets

KFJ_PACK_UNIT
Specifies the unit for the package if non-SMS/virtual tape is used.

KFJ_PACK_VOLUME
Specifies the volume for the package if non-SMS/virtual tape is used.

KFJ_PACK_MGMTCLAS
Specifies he management class for the package.

KFJ_PACK_STORCLAS
Specifies the storage class for the package.

KFJ_PACK_DATACLAS
Specifies the data class for the package. This is needed for large package files or tersed files as it will allow to specify multiple volumes.

- You can use the **OPTION NOUSS** parameter to bypass the z/OS UNIX System Services workflow stage and not include any related files or directories in the produced output data sets.

To run the **PACKAGE** action, use the following procedure.

Procedure

1. Update the KFJJMCM sample job in TKANSAM as follows (see example below):
   a) Select a **PACKAGE** action.
   b) Change the program name in the JCL **EXEC** statement from KCIOMEGA to KCIALPHA.
   c) Specify values for the required parameters **RTE_NAME** and **RTE_PLIB_HILEV**.
   d) (Optional) Specify any additional parameters as needed.
2. For an existing runtime environment, stop the started tasks for that runtime environment.
3. Make sure that any migrated SMS-managed data sets to be packaged have been recalled.
4. Run the KFJJMCM job to perform the packaging process and generate the related package files.
   Job messages for the **PACKAGE** action are written to the KCIPRINT SYSOUT data set.

Results

The packaging dump data sets and the metadata file are created.

Example

The following JCL creates DUMP data sets for the runtime environment that is defined by members of the TSOUID.MONSUITE.RTEDEF library.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1 EXEC PGM=KCIALPHA,REGION=0M,DYNAMNBR=256
//STEPLIB DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS DD *
ACTION PACKAGE
RTE_NAME RTE1
RTE_PLIB_HILEV TSOUID.MONSUITE

/*
```

Figure 31. Example JCL to perform the PACKAGE action

What to do next

Transfer the files to the target system using a method like FTPS, and run the **DEPLOY** action.

Related tasks

CREATE

The **CREATE** action creates an initial runtime environment definition that you can customize to match your requirements.

DEPLOY

The **DEPLOY** action deploys a packaged runtime environment to a remote system.

Remote deployment scenario

In a remote deployment scenario, you must create a runtime environment on a specific LPAR (the configuration LPAR), package the runtime environment data sets using the **PACKAGE** action, transfer the data sets to the remote target system (target LPAR), and deploy or restore the packaged runtime environment data sets on the target LPAR using the **DEPLOY** action.

DEPLOY

The **DEPLOY** action deploys a packaged runtime environment to a remote system.

Before you begin

The **DEPLOY** action can be run after you have successfully packaged a runtime environment using the **PACKAGE** action.

If you do not have any SMP/E target libraries on the system on which you want to run a **DEPLOY** action, you can use the utility, `TKANSAM(KFJMAINT)`, with the **BLDREMDS** action to build the necessary minimum data sets (TKANSAM, TKANMOD, and TKANCUS libraries) needed to run the action. Make sure you transfer the created data sets to your remote system where **DEPLOY** should run and where the necessary APF authorization of the TKANMOD library is made. These SMP/E target libraries are the minimum required to allow Configuration Manager to run for a full RTE. However, for a RTE that you are sharing with SMP/E, you will need to copy your entire set of SMP/E target libraries to the system where you run the **DEPLOY** action.

Important: When you use **DEPLOY**, the target runtime environment (RTE) data sets will be updated. Verify that these data sets are not in use before you use **DEPLOY**.

For more information about the complete remote deployment process, see “Remote deployment scenario” on page 137.

About this task

The **DEPLOY** action uses the dump data sets generated by the **PACKAGE** action to restore the runtime environment data sets on the target system. The **DEPLOY** action uses the DFSMSdss ADRDSSU program to restore the data sets. It performs an unconditional restore for the main VSAM and non-VSAM packages (fully replaces the data sets) and a conditional restore for the history packages (does not replace existing data sets). Note the following behaviors:

- If history dump data sets (`**.PACKHN` or `**.PACKHV`) are not found or failed to restore, a return code of 4 will be generated.
- If the main dump data sets (`**.PACKMN` or `**.PACKMV`) are not found or failed to restore, a return code of 8 will be generated.

Note: The **DEPLOY** action replaces all main package VSAM and non-VSAM files, but does not replace any history-related files. Therefore, it is normal for PACKHN and PACKHV deploy flows to end with RC=8. If you want to avoid this, for example if you roll out maintenance, do not transfer these packages to the target system. While normally a return code of 8 would cause Configuration Manager to stop, in this particular situation (history files), a return code of 8 is considered acceptable and will not prevent Configuration Manager from continuing to function.

When restoring (and potentially untersing) the packaged runtime environment, the **DEPLOY** action will reuse the following parameters used with the **PACKAGE** action:

- **KFJ_PACK_HILEV**
- **KFJ_ADRDSSU_ADMIN**
- **KFJ_PACK_DATACLAS**
- **KFJ_PACK_TERSE**
- **KFJ_PACK_UNIT**
- **KFJ_PACK_VOLUME**

See the "PACKAGE" on page 59 action for information about these parameters.

The following list provides more details about the **DEPLOY** action:

- The **PACKAGE** action requires the use of the KCIALPHA program. KCIALPHA is an APF-authorized version of KCIOMEGA.
- If **KFJ_PACK_UNIT** or **KFJ_PACK_VOLUME** is specified, it applies to all of the packages being untersed.
- For large packages being untersed, you should use **KFJ_PACK_DATACLAS** accordingly to allow multi-volume data set allocation for the extracted package files.
- If metadata file PACKMD is not available, packages are restored as is, retaining all high-level qualifiers and SMS properties.
- If **KFJ_PACK_TERSE** is set to Y, it first unterses the package. Untersed DMP file high-level qualifier and SMS parameters are used as specified in the following parameters:
  - **RTE_PLIB_HILEV**
  - **RTE_SMS_UNIT**
  - **RTE_SMS_VOLUME**
  - **RTE_SMS_MGMTCLAS**
  - **RTE_SMS_STORCLAS**
- If **RTE_SMS_VOLUME** is specified but **RTE_SMS_MGMTCLAS** is not, **RTE_SMS_MGMTCLAS** defaults to NULLMGMTCLAS. Similarly, **RTE_SMS_STORCLAS** defaults to NULLSTORCLAS.
- You can optionally control when to run the **DEPLOY** workflow stage that deploys the parts related to z/OS UNIX System Services. By default, the **DEPLOY** action automatically deploys files and directories related to z/OS UNIX, if they are present in the packaged runtime environment data sets. However, there might be cases where you would want to skip this step (for reasons such as authorization issues), and perform this step at a later time. You can use the **OPTION** parameter to specify the **NOUSS** or **USS** value to control this processing, as follows:

  NOUSS

  When **OPTION NOUSS** is specified, the **DEPLOY** action does not run the stage that deploys the parts related to z/OS UNIX.

  With this option, only z/OS data sets are deployed; files and directories related to z/OS UNIX are bypassed.

  USS

  When **OPTION USS** is specified, the **DEPLOY** action runs only the stage that deploys the parts related to z/OS UNIX; no other processing is performed.

  This option is useful when you want to refresh files and directories related to z/OS UNIX only.

To run the **DEPLOY** action, use the following procedure.

Procedure

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a **DEPLOY** action.

2. Change the program name in the JCL **EXEC** statement from KCIOMEGA to KCIALPHA.
3. Specify values for the required parameters **RTE_NAME** and **RTE_PLIB_HILEV**.
4. (Optional) Specify any additional parameters as needed.
5. Run the KFJJMCM job to perform the deploy process and restore the related package files.

   Job messages for the **DEPLOY** action are written to the KCIPRINT SYSOUT data set and to the $REPORT DD. If return code 4 or 8 is received, review the $REPORT DD statement in the JCL job output to ensure that the restore process completed successfully. For more information, see "DEPLOY action output" on page 63.

Example

The following JCL restores (*deploys*) the data sets for the runtime environment that has been packaged by the respective **PACKAGE** action, meaning the package that used RTE name RTE1 and TSOUID.MONSUITE.RTEDEF.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1 EXEC PGM=KCIALPHA,REGION=0M,DYNAMNBR=256
//STEPLIB DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS DD *
ACTION DEPLOY
RTE_NAME RTE1
RTE_PLIB_HILEV TSOUID.MONSUITE
/*
```

Figure 32. Example JCL to perform the DEPLOY action

Related tasks

PACKAGE
The **PACKAGE** action packages a runtime environment that can then be deployed to a remote system.

CREATE
The **CREATE** action creates an initial runtime environment definition that you can customize to match your requirements.

Remote deployment scenario
In a remote deployment scenario, you must create a runtime environment on a specific LPAR (the configuration LPAR), package the runtime environment data sets using the **PACKAGE** action, transfer the data sets to the remote target system (target LPAR), and deploy or restore the packaged runtime environment data sets on the target LPAR using the **DEPLOY** action.

DEPLOY action output

The **DEPLOY** action produces job output that provides information about the deployment of the data sets as a result of the job.

When running in default mode, the **DEPLOY** action writes summarized output and the job return code to the KCIPRINT SYSOUT data set and produces a more detailed report in the $REPORT DD. The following sections provide information about these output data sets and their contents, which includes data set deployment states and **DEPLOY** action return codes.

KCIPRINT

KCIPRINT contains a count of the data sets that the **DEPLOY** action processes, summarized by deployment state (*deployed*, *skipped*, *failed*), for each package data set. Looking at KCIPRINT, you can see if any data sets failed, but you will not see the exact data set names. This content is a condensed preview of what is included in the $REPORT output. KCIPRINT also contains the return code for the job.

The following figures show example KCIPRINT output for the **DEPLOY** action.

The example in Figure 33 on page 64 shows sample summary counts of the data sets processed by the **DEPLOY** action for each package (dump) data set; the actual number of data sets reported will be unique to the job:

```
IMB Z Monitoring Configuration manager is about to perform DEPLOY action

Processing: PACKMN
- Deployed: .. 30
- Skipped: ... 21
- Failed: .... 0

Processing: PACKHN
- Deployed: .. 0
- Skipped: ... 23
- Failed: .... 0

Processing: PACKMV
- Deployed: .. 38
- Skipped: ... 41
- Failed: .... 0

Processing: PACKHV
- Deployed: .. 0
- Skipped: ... 14
- Failed: .... 0

For more detailed information see $REPORT DD statement
```

Figure 33. DEPLOY action output in KCIPRINT — data set deployment summary

For more information about the deployment states, see “Data set deployment states” on page 65.

The example in Figure 34 on page 64 shows a possible return code for the **DEPLOY** action; the actual return code will be unique to the job:

```
KFU00004I KCIALPHA is ending; RC=4 SYSPLEX=sysplex LPAR=lpar DATE=…
```

Figure 34. DEPLOY action output in KCIPRINT — return code

For more information about return codes for the **DEPLOY** action, see “DEPLOY action return codes” on page 65.

For more information about KCIPRINT, see “How to navigate Configuration Manager action output” on page 147 and “Configuration Manager output data sets” on page 148.

$REPORT

$REPORT provides details about which data sets were deployed, which were intentionally skipped, and which failed. In addition, the report includes the deployment status of the z/OS UNIX System Services (z/OS UNIX) data sets as a result of the job.

Note: If you encounter issues with the **DEPLOY** action and the normal run does not explain what failed, use the **DEPLOY** action with **OPTION DEBUG** to print the complete output.

The following figures shows example $REPORT output.

The $REPORT content lists the individual data sets that were skipped, deployed, and failed (if any) for each package (dump) data set. The report sample in Figure 35 on page 65 shows a portion of the report for the PACKMN package data set only; the actual number of data sets reported will be unique to the job:

```
Processing data set: PACKMN
Will NOT REPLACE existing data sets
-----------------------------------------
Total deployed data sets: .......... 0
Total skipped data sets: ........... 21
Total failed to deploy data sets: .. 0

Details:
|- Skipped data sets:
||- rte_plib_hilev.rte_name.ims_id.RKOIPCSV
||- rte_plib_hilev.rte_name.ims_id.RKOIPFSV
||- rte_plib_hilev.rte_name.ims_id.RKOIPCSV
...
*****************************************

Processing data set: PACKMN
Will REPLACE existing data sets
-----------------------------------------
Total deployed data sets: .......... 30
Total skipped data sets: ........... 0
Total failed to deploy data sets: .. 0

Details:
|- Deployed data sets:
||- rte_plib_hilev.rte_name.EMBEDS
||- rte_plib_hilev.rte_name.PGMSCN
||- rte_plib_hilev.rte_name.PGMSDS
...
```

Figure 35. DEPLOY action output in $REPORT — data set deployment state

For more information about the deployment states, see “Data set deployment states” on page 65.

The report sample in Figure 36 on page 65 shows a portion of the report that lists the z/OS UNIX data sets that were deployed:

```
*****************************************
z/OS UNIX System Services:

Extracting files to: /proj/tdci2/MS/MSDEPL/PGMS
-----------------------------------------
KS3
KS3/rdm
KS3/rdm/jobs
KS3/rdm/jobs/RDMUNPX
...
```

Figure 36. DEPLOY action output in $REPORT — z/OS UNIX data sets

Data set deployment states

The **DEPLOY** action job output categorizes the processed data sets as follows:

Deployed
The **DEPLOY** action has successfully restored these data sets to the target system.

Skipped
The **DEPLOY** action skipped these data sets because they contain user data, such as configuration information and history-related files. These data sets should not be overwritten. It is normal to see numerous data sets skipped by the **DEPLOY** action.

Failed
The **DEPLOY** action did not restore these data sets to the target system. You should investigate any data sets that failed to deploy.

DEPLOY action return codes

The **DEPLOY** action uses the DFSMSdss ADRDSSU program to restore the data sets. It performs an unconditional restore for the main VSAM and non-VSAM packages (fully replaces the data sets) and a

conditional restore for the history packages (does not replace existing data sets). Note the following behaviors:

- If history dump data sets (**.PACKHN or **.PACKHV) are not found or failed to restore, a return code of 4 will be generated.
- If the main dump data sets (**.PACKMN or **.PACKMV) are not found or failed to restore, a return code of 8 will be generated.

The **DEPLOY** action replaces all main package VSAM and non-VSAM files, but does not replace any history-related files. Therefore, it is normal for PACKHN and PACKHV deploy flows to end with RC=8. If you want to avoid this, for example if you roll out maintenance, do not transfer these packages to the target system. While normally a return code of 8 would cause Configuration Manager to stop, in this particular situation (history files), a return code of 8 is considered acceptable and will not prevent Configuration Manager from continuing to function.

KCIOMEGA workflows

The KCIOMEGA program that runs IBM Z Monitoring Configuration Manager is a general-purpose job template engine. KCIOMEGA performs batch processing based on the job template that you specify. The processing that KCIOMEGA performs is known as a *workflow*.

Note: KCIALPHA is an APF-authorized version of KCIOMEGA. APF authorization is required for some actions.

In KCIOMEGA terms, Monitoring Configuration Manager is a workflow.

The KCIOMEGA program has two input data sets:

KCIFLOW
Contains a job template written in the KCIOMEGA *skeleton language*. The language is similar to a subset of JCL with additional syntax introduced by KCIOMEGA.

This job template is also known as a *workflow skeleton* or simply *skeleton*.

KCIOMEGA dynamically interprets the statements in the skeleton and performs the corresponding processing (the workflow).

Skeletons can invoke other skeletons, resulting in composite workflows that run sub-workflows.

The KCIOMEGA skeleton language is unpublished; not intended for users. However, it's human-readable plain text. If you're familiar with JCL syntax, the additional KCIOMEGA syntax is relatively straightforward to understand.

KCIVARS
Contains name-value pairs that set workflow variables.

KCIOMEGA replaces variable names in skeletons with the values from this data set.

Skeletons can refer to variable names in various contexts, such as data set names and "if ... then" conditions. Variables can determine the actions that workflows perform and the data sets that workflows use.

In the context of Monitoring Configuration Manager, workflow variables specify which action to perform and the location of the runtime environment definition on which you want to perform that action.

GENERATE and maintenance scenarios

The legacy PARMGEN uses several scenarios to perform maintenance actions. However, with Configuration Manager all of these maintenance scenarios are no longer needed, due to the GENERATE action.

The GENERATE action performs in one job what previously required multiple steps for the various maintenance scenarios. All of these scenarios are replaced with the GENERATE action. For more information, see “GENERATE” on page 45.

Parameters

In general, IBM Z Monitoring Configuration Manager and PARMGEN use the same parameters with the same default values. In a few cases, Monitoring Configuration Manager introduces a new parameter or sets a different default value for an existing parameter.

For parameters not described here, see Where to find information.

Parameters in the initial runtime environment configuration profile

The **CREATE** action creates an initial set of parameters that define a basic runtime environment. You can edit and add to this initial set to meet your specific requirements.

These parameters are also described in the OMEGAMON shared documentation or in the documentation for each product. The descriptions provided here include additional information to help you get started.

GBL_DSN_CICS_CTG_DLL

The CICS Transaction Gateway (TG) dynamic link library.

Required?

No

Default value

`SYS1.SCTGDLL`

Values

An MVS data set name.

GBL_DSN_CSF_SCSFMOD0

Configuration Manager by default uses the **GBL_DSN_CSF_SCSFMOD0** parameter, which requires the Integrated Cryptographic Service Facility (ICSF) to be available on the system. The ICSF load library contains the CSNB* modules used for password encryption. If you do not want to use ICSF, the **GBL_DSN_CSF_SCSFMOD0** parameter must be commented out or deleted.

This parameter is relevant only if ICSF is installed and configured on the z/OS system.

Required

Required if any of the following conditions apply:

- Password encryption is enabled for any components.
- A SOAP server is enabled on a remote monitoring server.
- Granular control of command requests is enabled (compatibility mode is disabled): the **KDS_KMS_SECURITY_COMPATMD** parameter is set to N.
- zAware feature is enabled for OMEGAMON® XE on z/OS.

Values

An MVS data set name.

Example

`CSF.SCSFMOD0`

GBL_DSN_DB2_DSNEXIT

The Db2 exit library.

The OMEGAMON collector uses the Db2 exit load modules in this library.

Required?

No

Values

An MVS data set name.

Example

`DSN.VC10.SDSNEXIT`

GBL_DSN_DB2_LOADLIB_Vn

The load library for the version of Db2 that your site is running.

In the parameter name, **n** is the Db2 version number. For example, **GBL_DSN_DB2_LOADLIB_V12**.

Specify a **GBL_DSN_DB2_LOADLIB_Vn** parameter for each Db2 version that you want to monitor.

Required

Required if the runtime environment contains the Db2 monitoring agent.

Values

An MVS data set name.

Example

`DSN.VC10.SDSNLOAD`

GBL_DSN_DB2_RUNLIB_Vn

The run library for the version of Db2 that your site is running.

In the parameter name, **n** is the Db2 version number. For example, **GBL_DSN_DB2_RUNLIB_V12**.

Specify a **GBL_DSN_DB2_RUNLIB_Vn** parameter for each Db2 version that you want to monitor. The library should contain the modules `DSNTIAD` and `DSNTIAUL` to be used to run in batch.

IBM Z Monitoring Configuration Manager uses the library to generate **GRANT** and **BIND** jobs that prepare the Db2 subsystems for monitoring.

Required

Required if the runtime environment contains the Db2 monitoring agent.

Values

An MVS data set name.

Example

DSN.VC10.RUNLIB.LOAD

GBL_DSN_IMS_RESLIB

The IMS SDFSRESL library.

Description

The IMS SDFSRESL library contains the CQSREG00 action module required to enable the Common Queue Server (CQS). The CQS and shared queues allow users to take advantage of the Parallel Sysplex® environment.

Note: The DISCOVER action of IBM Z® Monitoring Configuration Manager discovers the value of the KI2_I1nn_CLASSIC_IMS_RESLIB parameter, which also specifies an IMS SDFSRESL library. Depending on how IMS is configured at your site, the same value might be appropriate for GBL_DSN_IMS_RESLIB.

Required or optional
Required if the runtime environment configures the IMS monitoring agent.

Default value
IMS.SDFSRESL

GBL_DSN_IMS_SCEXLINK

The IMS Connect product load library.

The IMS monitoring agent uses the IMS Connect Extensions Publisher API. The agent requires the IMS Connect Extensions product and functional support load libraries to connect to and collect performance and statistics data from the IMS Connect address space.

Required

Required if the runtime environment configures the IMS monitoring agent.

Default value

IMS.SCEXLINK

Values

An MVS data set name.

GBL_DSN_IMS_SFUNLINK

The IMS Connect functional support load library.

The IMS monitoring agent uses the IMS Connect Extensions Publisher API. The agent requires the IMS Connect Extensions product and functional support load libraries to connect to and collect performance and statistics data from the IMS Connect address space.

Required

Required if the runtime environment configures the IMS monitoring agent.

Default value

IMS.SFUNLINK

Values

An MVS data set name.

GBL_DSN_WMQ_SCSQANLE

The IBM® MQ language library.

Required

Required if the runtime environment configures the MQ monitoring agent.

Default value

`CSQ.V9R0M0.SCSQANLE`

Values

An MVS data set name.

GBL_DSN_WMQ_SCSQAUTH

The IBM MQ authorized load library.

Required

Required if the runtime environment configures the MQ monitoring agent.

Default value

`CSQ.V9R0M0.SCSQAUTH`

Values

An MVS data set name.

GBL_DSN_NETVIEW_CNMLINK

Identifies the IBM Z NetView CNMLINK library

Required

It is required only if using an IBM Z NetView Agent.

Default value

`NETVIEW.VNRNMN.CNMLINK`

Values

An MVS data set name.

GBL_HFS_JAVA_DIRn

The z/OS UNIX System Services path of the Java home directory.

The path consists of the concatenated values of the **GBL_HFS_JAVA_DIR1** and **GBL_HFS_JAVA_DIR2** parameters.

Typically, you only specify **GBL_HFS_JAVA_DIR1**. **GBL_HFS_JAVA_DIR2** is provided as a convenience to specify the remainder of a long directory path.

Required

Required if you are enabling the self-describing agent (SDA) functionality in the z/OS monitoring server (TEMS) and agents.

Default value

GBL_HFS_JAVA_DIR1 /usr/lpp/java/IBM/J8.0_64

GBL_HFS_JAVA_DIR2 *none* (empty string)

Runtime members

See the **TEMS_JAVA_BINPATH** parameter in the KDSDPROF member of the RKANDATV library.

Values

A z/OS UNIX directory path. Must begin with a forward slash (/).

Do not specify a trailing /bin directory in the value. IBM Z Monitoring Configuration Manager appends /bin to the value that you specify.

Example

If **GBL_HFS_JAVA_DIR1** is set to /my/own/copy and **GBL_HFS_JAVA_DIR2** is set to /of/java, then the resulting directory path is a concatenation of these two values: /my/own/copy/of/java.

GBL_TARGET_HILEV

The high-level qualifiers of the target libraries, such as TKANDATV and TKANMOD.

The **CREATE** action of IBM Z Monitoring Configuration Manager sets the **GBL_TARGET_HILEV** parameter to the high-level qualifiers of the data set name specified by the KCIFLOW DD statement in the job step that performs the **CREATE** action.

Required?

Yes

Values

MVS data set high-level qualifiers.

Example

If the KCIFLOW DD statement of the job step that performs the **CREATE** action specifies the data set name MONSUITE.TKANCUS, then the **CREATE** action sets the value of **GBL_TARGET_HILEV** to MONSUITE.

GBL_USS_TKANJAR_PATH

The path of the z/OS UNIX System Services directory that the SMP/E installation jobs define using the ddname TKANJAR.

Depending on your local site practices, this path might refer to a copy, rather than the original SMP/E-managed directory.

Required or optional

Required if the runtime environment configures either of the following monitoring agents:

- CICS Transaction Gateway (TG). The corresponding configuration parameter is **CONFIGURE_CICS_TG_KGW**.

- Java Virtual Machine (JVM). The corresponding configuration parameter is **CONFIGURE_JVM_KJJ**.

Default value
`/usr/lpp/kan/bin/IBM`

Permissible values

z/OS UNIX directory path. Must begin with a forward slash (/).

Related parameters
GBL_DSN_SYS1_SBPXEXEC

RTE_USS_RTEDIR

RTE_USS_MKDIR_MODE

RTE_NAME

The runtime environment name.

IBM Z Monitoring Configuration Manager uses this name for various purposes, including:

- MVS member names
- MVS data set name qualifiers
- z/OS UNIX System Services directory name, all uppercase

Required?

Yes

Values

1 - 8 characters.

Example

RTE1

Related reference
Runtime members
The **GENERATE** action generates runtime members in locations (MVS data set names and z/OS UNIX System Services paths) that are specified by parameters.

RTE_PLIB_HILEV
The default high-level qualifiers of runtime members that are stored in MVS data sets.

RTE_USS_RTEDIR
The path where runtime members are stored in z/OS UNIX System Services.

RTE_PLIB_HILEV

The default high-level qualifiers of runtime members that are stored in MVS data sets.

The **RTE_PLIB_HILEV** parameter sets the default values of several parameters that specify runtime member locations:

RTE_HILEV
RTE_VSAM_HILEV
GBL_DSN_SYS1_PROCLIB
GBL_DSN_SYS1_VTAMLIB
GBL_DSN_SYS1_VTAMLST

Required?

Yes

Values

To avoid exceeding the z/OS 44-character limit for data set names, the combined length of **RTE_NAME** and **RTE_PLIB_HILEV** should not exceed 28 characters. For example, if **RTE_NAME** is 8 characters, then **RTE_PLIB_HILEV** should not exceed 20 characters.

Related reference

RTE_NAME
The runtime environment name.

Runtime members
The **GENERATE** action generates runtime members in locations (MVS data set names and z/OS UNIX System Services paths) that are specified by parameters.

RTE_USS_RTEDIR
The path where runtime members are stored in z/OS UNIX System Services.

RTE_SECURITY_CLASS

This parameter specifies a System Authorization Facility (SAF) security class name for OMEGAMON enhanced 3270 user interface security controls.

Description

Use this parameter to specify the SAF security class for OMEGAMON enhanced 3270 user interface (enhanced 3270UI) security controls. The enhanced 3270UI performs security validation processing by authenticating the user identity using the SAF interface. The existence of the SAF user and its validity (that is, whether it is suspended) are always checked.

This parameter applies to the OMEGAMON enhanced 3270 user interface and the OMEGAMON monitoring agents that use the enhanced 3270UI. Individual products have additional SAF security settings that are specific to the respective product (for example, how to secure product-specific Take Action requests). To secure other products, see the product-specific documentation for information.

Important (for Configuration Manager users only): If a value is not specified for override parameter K*pp*_SECURITY_ACTION_CLASS (where *pp* is C5, M5, or N3), then the RTE_SECURITY_CLASS parameter value will be assigned as the default value.

Required or optional
Optional

Default value
None

Permissible values
A valid SAF class name, which can be a string of up to 8 characters. If you are using ACF2 as your external security resource manager, specify a maximum of 3 characters.

Related parameters

- KOB_SAF_ACTION_CLASS_NAME
- KC5_SECURITY_ACTION_CLASS
- KM5_SECURITY_ACTION_CLASS
- KN3_SECURITY_ACTION_CLASS

RTE_SECURITY_FOLD_PASSWORD_FLAG

Folds passwords to uppercase.

By default, TMS:Engine folds logon passwords to uppercase. However, IBM RACF® V1.7 and later supports mixed-case passwords.

If you want to use mixed-case passwords and if all your monitoring agents support them, set the **RTE_SECURITY_FOLD_PASSWORD_FLAG** to N.

If any of your monitoring agents do not support mixed-case passwords, do not activate the SETROPTS PASSWORD(MIXEDCASE) option in RACF and do not enable mixed-case passwords in your runtime environments. Use the default value of Y for this parameter.

Required?

No

Default value

Y

Values

Y
Fold passwords to uppercase.

N
Do not fold password to uppercase. Allow mixed-case passwords.

RTE_SECURITY_USER_LOGON

The security system to be used for the runtime environment.

If you specify a security system, verify that it is installed and configured correctly for your site.

The **RTE_SECURITY_USER_LOGON** parameter specifies which system will be used to validate users signing on to the Tivoli Enterprise Portal (TEP), but it does not enable validation. To enable validation of users signing on to TEP, the **KDS_TEMS_SECURITY_KDS_VALIDATE** parameter value must be Y (its default value).

Required?

No

Default value

NONE

Values

NONE
No security.

RACF
IBM z/OS Security Server.

ACF2
CA ACF2.

If you specify ACF2, you must set the **GBL_DSN_ACF2_MACLIB** parameter to the name of the ACF2 macro library.

TSS
CA Top Secret.

NAM
Network Access Manager.

SAF
IBM z/OS System Authorization Facility API.

RTE_STC_PREFIX

The prefix of started task names for this runtime environment.

Required?

No

Default value

`IBM`

Values

1 - 4 characters.

Example

The **CREATE** action of IBM Z Monitoring Configuration Manager sets the following value:

`OMEG`

(an abbreviation of OMEGAMON).

RTE_TCP_HOST

This parameter contains the TCP/IP host name or dotted decimal IPV4 address of the z/OS® system where the runtime environment is being defined.

The RTE_TCP_HOST parameter sets the default values of several parameters that specify the host name to ensure consistent settings for all agents/products in the runtime environment, including:

- KDS_TEMS_TCP_HOST
- Kpp_TEMS_TCP_HOST

Required?

No

Default value

`&SYSIPHOSTNAME`

Values

1 - 32 characters

RTE_TCP_PORT_NUM

The port number for communication over IP.

The **RTE_TCP_PORT_NUM** parameter sets the default values of several parameters that specify port numbers, including:

KDS_TEMS_TCP_PIPE_PORT_NUM
Kpp_TEMS_TCP_PIPE_PORT_NUM

Required?

No

Default value

1918

RTE_TEMS_NAME_NODEID

Identifies the monitoring server for internal processing.

The **KDS_HUB_TEMS_NAME_NODEID** parameter of remote monitoring servers must refer to the **RTE_TEMS_NAME_NODEID** of the hub monitoring server. For example, if the hub sets **RTE_TEMS_NAME_NODEID** to HUB:TEMS, then the runtime environments for remote monitoring servers must set **KDS_HUB_TEMS_NAME_NODEID** to HUB:TEMS.

Required?

No

Default value

rte_name:CMS

Example

The **CREATE** action of IBM Z Monitoring Configuration Manager sets the following value:

rte_name:TEMS

where TEMS stands for Tivoli Enterprise Monitoring Server, reflecting current terminology.

RTE_USS_RTEDIR

The path where runtime members are stored in z/OS UNIX System Services.

The runtime environment name, parameter **RTE_NAME**, is appended to the value of **RTE_USS_RTEDIR** as a directory name.

The TSO user ID that runs IBM Z Monitoring Configuration Manager jobs must have permission to write to this directory, otherwise the **GENERATE** action will fail.

Required?

No

Default value

/var/rtehome

Example

If **RTE_USS_RTEDIR** is /var/rtehome and **RTE_NAME** is RTE1, then runtime members are stored in:

/var/rtehome/RTE1/*

Related reference
RTE_NAME

The runtime environment name.

RTE_PLIB_HILEV
The default high-level qualifiers of runtime members that are stored in MVS data sets.

Runtime members
The **GENERATE** action generates runtime members in locations (MVS data set names and z/OS UNIX System Services paths) that are specified by parameters.

RTE_VTAM_APPLID_PREFIX

The global VTAM applid prefix to be used to build the VTAM applids for products in this runtime environment.

Required?

No

Default value

CTD

Example

The **CREATE** action of IBM Z Monitoring Configuration Manager sets the following value:

OM*xx*

where *xx* is the value of the z/OS static system symbol **SYSCLONE**. **SYSCLONE** is a 1- or 2-character shorthand notation for the system (LPAR) name.

Parameters with significant default values

The runtime environment defined by the initial set of parameters is configured not just by the relatively small number of parameters in that set, but also by the default values of many other parameters.

The following parameter is not included in the initial set, but its default value significantly affects the runtime environment.

RTE_TYPE

Determines whether runtime members are a full stand-alone set or shared with SMP/E target installation libraries.

Required?
No

Default value

Configuration Manager: SHARING

PARMGEN: FULL

Values
The following descriptions apply to Configuration Manager:

FULL
Stand-alone runtime members. Runtime members have no dependency on target libraries.

SHARING
Some runtime members refer to the target libraries.

The high-level qualifiers of the target libraries are specified by the **GBL_TARGET_HILEV** parameter.

SHARING reduces the storage requirement for each runtime environment.

If **RTE_TYPE** is SHARING, then the value of the RTE_SHARE parameter must be SMP.

Parameters with different default values than PARMGEN

IBM Z Monitoring Configuration Manager sets some different default parameter values than PARMGEN.

In some cases, instead of changing the default value of a parameter, Monitoring Configuration Manager sets a different example value in the initial set of parameters.

Table 8. Parameters with different default values in PARMGEN and Monitoring Configuration Manager

Parameter	PARMGEN default value and reason for change	Monitoring Configuration Manager default value
GBL_DSN_CSF_SCSFMOD	In PARMGEN, this parameter is commented out in the WCONFIG file. In Configuration Manager, it is explicitly added in the GBL$PARM member that is generated after the **CREATE** or **MIGRATE** action, as it is relevant for several security-related aspects of the product configuration (such as password encryption). As described in the parameter description, if your installation does not use the Integrated Cryptographic Service Facility (ICSF), you can remove or comment out this parameter in your RTEDEF(GBL$PARM) or RTEDEF(GBL$*lpar*).	CSF.SCSFMOD0
GBL_DSN_SYS1_PROCLIB **GBL_DSN_SYS1_VTAMLIB** **GBL_DSN_SYS1_VTAMLST**	SYS1.PROCLIB SYS1.VTAMLIB SYS1.VTAMLST Sites typically have their own procedures for copying members to system libraries, limited to z/OS system administrators with special permissions.	*rte_hilev*.SYS1.PROCLIB *rte_hilev*.SYS1.VTAMLIB *rte_hilev*.SYS1.VTAMLST
KDS_KMS_SDA	N Using the self-describing agent (SDA) function is best practice.	Y
KMQ_HISTCOLL_DATA_FLAG	N Collecting historical data is best practice.	Y
KMQ_STARTMON_ACTIVEONLY	NO Only monitoring active queue managers is best practice.	YES

Table 8. Parameters with different default values in PARMGEN and Monitoring Configuration Manager (continued)

Parameter	PARMGEN default value and reason for change	Monitoring Configuration Manager default value
KYN_XAI01_SUBAGENT_ JAVAHOME	`/usr/lpp/java/J7.1` There is no need for a default value specifically for this subagent. Use the existing global parameters as a default value instead.	Concatenation of the following two parameter values: *gbl_hfs_java_dir1* *gbl_hfs_java_dir2*
RTE_TYPE	FULL SHARING reduces the storage requirement for each runtime environment.	SHARING **Note:** When **RTE_TYPE** is SHARING, then the value of the **RTE_SHARE** parameter must be SMP, which indicates sharing with SMP/E target libraries. Configuration Manager does not use base or sharing-with-base runtime environments (as in PARMGEN).
RTE_USS_RTEDIR	`/rtehome` Creating a new subdirectory of the root directory is bad practice.	`/var/rtehome`
RTE_X_SECURITY_EXIT_LIB	*rte_hilev.rtename*.RKANSAMU Identifies the security exits library currently used in PARMGEN.	*rte_hilev.rte_name* .SECEXITS

Parameters introduced by Monitoring Configuration Manager

IBM Z Monitoring Configuration Manager introduces some parameters that do not exist in PARMGEN.

Global (GBL) parameter introduced by Configuration Manager

The global parameters provide default settings for installation and common system library names.

The global parameter introduced by IBM Z Monitoring Configuration Manager is explained in this section.

GBL_UTIL_BINDER

This parameter allows you to override the default binder program name.

Description

To use a binder program other than the default IEWL, specify the name of the program in this parameter. If the name is in the list of documented IEWL aliases, then KCIOMEGA will rename the SYSPRINT, as it does for IEWL.

Required or optional
Required

Default value
IEWL

Permissible values
A valid MVS program name

Runtime environment (RTE) parameters introduced by Configuration Manager

The runtime environment parameters provide configuration settings for an individual runtime environment and default settings for the OMEGAMON components and products configured in that runtime environment.

The runtime environment parameters introduced by IBM Z Monitoring Configuration Manager are explained in this section. The **RTE_COMM_PROTOCOLn**, **RTE_TCP_*****, and **RTE_VTAM_NETID** parameters offer an easy way to set all components to the same values, rather than setting parameters individually for each component.

RTE_COMM_PROTOCOLn

Sets the communication protocol choices of all components in the runtime environment.

The **RTE_COMM_PROTOCOLn** (**n**: 1 - 7) parameters set the value of the **KDS_TEMS_COMM_PROTOCOLn** and **Kpp_AGT_COMM_PROTOCOLn** parameters.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

Required?

No

Default value

none

Values

Value	Value in runtime member RKANPARU (K*pp*ENV)	Protocol description	Corresponding protocol-specific parameters
IPPIPE	IP.PIPE	Non-secure TCP over IPv4	**RTE_TCP_HOST** **RTE_TCP_PORT_NUM**
IP	IP.UDP	Non-secure UDP over IPv4	**RTE_TCP_HOST** **RTE_TCP_UDP_PORT_NUM**
IP6PIPE	IP6.PIPE	Non-secure TCP over IPv6	**RTE_TCP_HOST** **RTE_TCP_PIPE6_PORT_NUM**
IP6	IP6.UDP	Non-secure UDP over IPv6	**RTE_TCP_HOST** **RTE_TCP_UDP6_PORT_NUM**
IPSPIPE	IP.SPIPE	Secure (SSL/TLS) TCP over IPv4	**RTE_TCP_HOST** **RTE_TCP_PIPES_PORT_NUM**
IPS6PIPE	IP6.SPIPE	Secure (SSL/TLS) TCP over IPv6	**RTE_TCP_HOST** **RTE_TCP_PIPE6S_PORT_NUM**
SNA	SNA.PIPE	NCS RPC: Systems Network Architecture implementation of the Network Computing System Remote Procedure Call API	**Kpp_TEMS_VTAM_APPL_LLB_BROKER** **Kpp_TEMS_VTAM_LU62_DLOGMOD** **Kpp_TEMS_VTAM_LU62_MODETAB** **RTE_VTAM_NETID**

Default values of **KDS_TEMS_COMM_PROTOCOLn** and **Kpp_AGT_COMM_PROTOCOLn**:

n	Value
1	IPPIPE
2	SNA

Example

Parameter	Value	Description
RTE_COMM_PROTOCOL1	IPS6PIPE	First choice: secure TCP over IPv6
RTE_COMM_PROTOCOL2	IPSPIPE	Second choice: secure TCP over IPv4
RTE_COMM_PROTOCOL3	IP6PIPE	Third choice: non-secure TCP over IPv6
RTE_COMM_PROTOCOL4	IPPIPE	Fourth choice: non-secure TCP over IPv4

Related reference

Communication between monitoring components
In a typical topology, monitoring agents communicate with remote monitoring servers, and remote monitoring servers communicate with a single, central hub monitoring server.

RTE_TCP_KDEB_INTERFACELIST

Directs all components in the runtime environment to connect to a specific TCP/IP local interface.

The **RTE_TCP_KDEB_INTERFACELIST** parameter sets the value of the **KDS_TEMS_TCP_KDEB_INTERFACELIST** and **Kpp_AGT_TCP_KDEB_INTERFACELIST** parameters.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

If the z/OS image has more than one TCP/IP interface or network adapter, you can use this parameter to direct components to connect to a specific TCP/IP local interface.

Required?

No

Default value

!* (exclamation point followed by an asterisk)

Runtime members

See the K*pp*ENV member of the RKANPARU library.

Values

Character string, maximum length 44, specifying one or more network interfaces to use.

To set a network interface list, supply one of the following values:

- The host name or IP address of the preferred interface.
- A list of host names or IP addresses, in descending order of preference. Use a blank (space) to separate the entries.
- An asterisk (*) to prefer the interface associated with the default host name for the z/OS image. To display this value, enter TSO HOMETEST at the command line.
- An exclamation point followed by an asterisk (!*) to use only the interface associated with the default host name for the z/OS image.

- An exclamation point followed by a host name or IP address (!*hostname*) to use only the interface associated with *hostname*.

Note:

- If you set the value of this parameter to !* or !*hostname*, you must specify the same value for every component and product configured in all runtime environments on the same z/OS image.
- In the default character set (LANG=en_US.ibm-037), the code for an exclamation point is x'5A'. If you are using a character set other than the default, a different character might map to that code. To require a specific network interface, use the character that maps to x'5A' in your character set.

For a high-availability hub, specify the value of this parameter as !*dvipa_hostname*, where *dvipa_hostname* is the private DVIPA name set for the **KDS_TEMS_TCP_HOST** parameter.

RTE_TCP_PIPE6_PORT_NUM

Sets the port number for all components in the runtime environment that use the TCP over IPv6 communication protocol.

The **RTE_TCP_PIPE6_PORT_NUM** parameter sets the value of the **KDS_TEMS_TCP_PIPE6_PORT_NUM** and **Kpp_TEMS_TCP_PIPE6_PORT_NUM** parameters.

This parameter is used only if one of the parameters that sets communication protocol choices, ***_COMM_PROTOCOLn**, specifies the value for this protocol, IP6PIPE.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

Required?

No

Default value

none

Values

A port number.

RTE_TCP_PIPE6S_PORT_NUM

Sets the port number for all components in the runtime environment that use the secure TCP over IPv6 communication protocol.

The **RTE_TCP_PIPE6S_PORT_NUM** parameter sets the value of the **KDS_TEMS_TCP_PIPE6S_PORT_NUM** and **Kpp_TEMS_TCP_PIPE6S_PORT_NUM** parameters.

This parameter is used only if one of the parameters that sets communication protocol choices, ***_COMM_PROTOCOLn**, specifies the value for this protocol, IPS6PIPE.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

Required?

No

Default value

none

Values

A port number.

RTE_TCP_PIPES_PORT_NUM

Sets the port number for all components in the runtime environment that use the secure TCP over IPv4 communication protocol.

The **RTE_TCP_PIPES_PORT_NUM** parameter sets the value of the **KDS_TEMS_TCP_PIPES_PORT_NUM** and **Kpp_TEMS_TCP_PIPES_PORT_NUM** parameters.

This parameter is used only if one of the parameters that sets communication protocol choices, ***_COMM_PROTOCOLn**, specifies the value for this protocol, IPSPIPE.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

Required?

No

Default value

none

Values

A port number.

RTE_TCP_STC

Sets the TCP/IP stack for all components in the runtime environment that use the IP communication protocol.

The **RTE_TCP_STC** parameter sets the value of the **KDS_TEMS_TCP_STC** and **Kpp_AGT_TCP_STC** parameters.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

Required?

No

Default value

(pound or hash sign)

Runtime members

Sets the value of the **TCP/IP_USERID** parameter in the K*pp*INTCP member of the RKANPARU library.

Setting **RTE_TCP_STC** to # (pound or hash sign) sets the value of **TCP/IP_USERID** to a blank (space), which allows TCP/IP to decide the stack associated with the address space, for better load balancing.

Values

If the LPAR contains more than one TCP/IP stack, specify the started task name of the TCP/IP stack you want to use. Alternatively, specify a hash sign (#), which is translated to a blank and allows the TCP/IP environment to choose the stack to use, either through TCP/IP definitions or through the use of the SYSTCPD DD statement.

Whichever method is used to select a TCP/IP stack in a multi-stack environment, the Tivoli® Management Services components continue to use that stack, even if a different stack becomes the primary stack. Therefore, in a multi-stack environment, it is best to specify the started task name of the TCP/IP stack to be used, rather than specifying a wildcard or a blank.

RTE_TCP_UDP_PORT_NUM

Sets the port number for all components in the runtime environment that use the UDP over IPv4 communication protocol.

The **RTE_TCP_UDP_PORT_NUM** parameter sets the value of the **KDS_TEMS_TCP_UDP_PORT_NUM** and **Kpp_TEMS_TCP_UDP_PORT_NUM** parameters.

This parameter is used only if one of the parameters that sets communication protocol choices, ***_COMM_PROTOCOLn**, specifies the value for this protocol, IP.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

Required?

No

Default value

none

Values

A port number.

RTE_TCP_UDP6_PORT_NUM

Sets the port number for all components in the runtime environment that use the UDP over IPv6 communication protocol.

The **RTE_TCP_UDP6_PORT_NUM** parameter sets the value of the **KDS_TEMS_TCP_UDP6_PORT_NUM** and **Kpp_TEMS_TCP_UDP6_PORT_NUM** parameters.

This parameter is used only if one of the parameters that sets communication protocol choices, ***_COMM_PROTOCOLn**, specifies the value for this protocol, IP6.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

Required?

No

Default value

none

Values

A port number.

RTE_VTAM_NETID

Sets the VTAM network ID for all components in the runtime environment that use the SNA communication protocol.

Note: This parameter is valid in both PARMGEN and Configuration Manager; however, its function differs slightly. This topic describes this parameter as it applies to Configuration Manager.

The **RTE_VTAM_NETID** parameter sets the value of the **KDS_TEMS_VTAM_NETID** and **Kpp_TEMS_VTAM_NETID** parameters.

This parameter is used only if one of the parameters that sets communication protocol choices, ***_COMM_PROTOCOLn**, specifies the value for this protocol, SNAPIPE.

This parameter offers an easy way to set all components to the same values, rather than setting parameters individually for each component.

Required?

No

Default value

none

Values

A VTAM network ID.

RTE_X_OVERRIDE_EMBEDS_LIB

This parameter specifies the name of the source library for override embed members.

Override embed files can be used to add user-defined parameters and values that might be overwritten during the **GENERATE** action.

If the **KFJ_USE_EMBEDS** parameter is set to Y, this parameter is automatically defined in the *rte_plib_hilev*.RTEDEF(*rte_name*) member during an initial **CREATE** or **MIGRATE** action when creating an RTE.

Note: If the **KFJ_USE_EMBEDS** parameter was not set to Y when creating the RTE, you must manually add this parameter to the *rte_plib_hilev*.RTEDEF(*rte_name*) member to enable the use of override embed members for the RTE.

This library contains all the override embed members for all products installed in the respective CSI.

Required?

No

Default value

<rte_plib_hilev>.<rte_name>.EMBEDS

Values

A valid MVS data set name.

Example

TEST1.TST.DEMO.EMBEDS

Related parameters

KFJ_USE_EMBEDS
KFJ_EMBEDS_LIB

RTE_X_SECURITY_EXIT_LIB

This parameter specifies the library containing the security exits used for the runtime environment.

Note: This parameter is valid in both PARMGEN and Configuration Manager; however, its function differs slightly. This topic describes this parameter as it applies to Configuration Manager.

The **RTE_X_SECURITY_EXIT_LIB** parameter specifies the name of the global runtime environment library that contains all of the OMEGAMON and IBM Tivoli Monitoring-related product security exits (such as KOBSUPDT OMEGAMON K*pp*SUPDI exits, Tivoli Monitoring Services: Engine security exits, external security exits).

Required?

No

Default value

<rte_plib_hilev>.<rte_name>.SECEXITS

Values

A valid data set name.

Example

TEST1.TST.DEMO.SECEXITS

Configuration Manager (KFJ) parameters

The Configuration Manager parameters have the prefix KFJ.

KFJ_ADRDSSU_ADMIN

Specifies the ADRSSU administrator parameter.

The KFJ_ADRDSSU_ADMIN parameter works as a switch during PACKAGE and DEPLOY actions:

- Y – Enables ADMINISTRATOR mode while using the ADRDSSU program when creating or restoring DUMP data sets.
- N – Disables ADMINISTRATOR mode while using the ADRDSSU program when creating or restoring DUMP data sets.

Use KFJ_ADRDSSU_ADMIN according to the potential security considerations at your site. This parameter is not stored in any RTEDEF member and will be specified in the KCIVARS DD of your Configuration Manager batch JCL.

Required or optional
Required

Default value
N

Permissible values
Y, N

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_EMBEDS_LIB

This parameter identifies the data set that contains the override embed values for the RTE.

Description

Use this parameter to specify the name of the data set (the *embeds data set*) that will contain the override embed values to use for the RTE. The data set must be accessible by Monitoring Configuration Manager. An example of the data set name is *highlevel*.CFM.RTEDEF.EMBEDS.

This parameter is active only if **KFJ_USE_EMBEDS** is set to Y.

The default name of the embeds data set is *rte_plib_hilev*.*rte_name*.EMBEDS, which is intended for a single RTE. By using **KFJ_EMBEDS_LIB**, you can customize the name of the embeds data set and use the same override embed parameters and values for multiple RTEs. If you will be using different override values for different RTEs, consider using the *rte_name* or *sysname* values in the data set name.

If the specified data set does not exist when the **CREATE** or **MIGRATE** action runs, Monitoring Configuration Manager creates the data set and populates it with the override embed parameters and values for the products that are installed in the respective CSI used to build the RTE. For the **MIGRATE** action, this library will contain the override embed parameters and values from the RTE being migrated from.

If the specified data set exists when the **CREATE** or **MIGRATE** action runs, none of the existing members in it are changed or removed; only new members are added, which can occur if the CSI used to build the RTE has changed to add products.

Required or optional
Optional

Default value
rte_plib_hilev.*rte_name*.EMBEDS

Permissible values
A valid MVS data set name

Example
TEST1.TST.DEMO.EMBEDS

Related parameters
KFJ_USE_EMBEDS

RTE_X_OVERRIDE_EMBEDS_LIB

KFJ_LOCAL_HILEV

Assigns a local non-VSAM high-level qualifier.

The KFJ_LOCAL_HILEV parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member, and identifies the non-VSAM high-level qualifier to be used for allocating the local runtime data sets when the GENERATE action is run. The names of the non-VSAM data sets will be generated by appending the appropriate suffix to this parameter.

KFJ_LOCAL_HILEV maps to the local value of RTE_HILEV that is used on the deployment target system.

Required or optional
Optional

Default value
%KFJ_LOCAL_PLIB_HILEV%

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_KD5_RUN_ALLOC

This parameter controls if the job for Db2-related data set allocation is submitted. This parameter is used for remote deployment scenarios.

Description

In a remote deployment scenario, you can use this parameter to specify whether to submit the Db2-related data set allocation job. This job allocates all operational data sets required for the enabled functions (for example, to collect data for Thread History). This job does not overwrite existing operational data sets.

Specify one of the following values:

GENERATE
Trigger the ALLOCDS JCL job during the GENERATE action. With this option, you cannot customize the KD2 operational data set allocation parameters.

DEPLOY
Trigger the ALLOCDS JCL job during the DEPLOY action. With this option, you can customize the KD2 operational data set allocation parameters.

NONE
Do not trigger the ALLOCDS JCL job. With this option, you can customize the KD2 operational data set allocation parameters.

Tip: For non-remote deployment scenarios (PACKAGE, DEPLOY), you can use parameter KD2_OMPE_RUNALLOC to perform the same function.

Required or optional
Optional

Default value
GENERATE

Permissible values
GENERATE, DEPLOY, NONE

Related parameters
KD2_OMPE_RUNALLOC

KFJ_LOCAL_PDS_HILEV

Specifies the PDS V1 local high-level qualifier.

The KFJ_LOCAL_PDS_HILEV parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member, and identifies the high-level qualifier for the local Persistent Data Store (PDS) V1 data sets. It maps to the local value of RTE_PDS_HILEV that is used on the deployment target system.

Required or optional
Optional

Default value
`%KFJ_LOCAL_HILEV%.%RTE_NAME%`

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_PLIB_HILEV

Specifies the high-level qualifier for local libraries.

The KFJ_LOCAL_PLIB_HILEV parameter is used to specify the local high-level qualifier for the generated RTEDEF/EMBEDS and SECEXITS data sets. The parameter is specified in the KCIVARS DD of the Configuration Manager batch JCL. Specifying this parameter will trigger the generation of member RTEDEF(PCK$PARM).

This parameter is used to populate default values for other KFJ_LOCAL_* high-level qualifier parameters in RTEDEF(PCK$PARM), such as:

- KFJ_LOCAL_HILEV
- KFJ_LOCAL_VSAM_HILEV

Required or optional
Optional

Default value
N/A

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_SMS_MGMTCLAS

Identifies the MGMTCLAS for local non-VSAM libraries.

The KFJ_LOCAL_SMS_MGMTCLAS parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member and indicates the SMS Management Class to be used when allocating the local non-VSAM runtime data sets. KFJ_LOCAL_SMS_MGMTCLAS maps to the local value of RTE_SMS_MGMTCLAS that is used on the deployment target system.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_SMS_STORCLAS

Identifies the STORCLAS for local non-VSAM libraries.

This parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member, and defines the SMS Storage Class to be used when allocating the local non-VSAM runtime data sets. KFJ_LOCAL_SMS_STORCLAS maps to the local value of RTE_SMS_STORCLAS that is used on the deployment target system.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_SMS_UNIT

Specifies the Unit name for local non-VSAM libraries.

The KFJ_LOCAL_SMS_UNIT parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member, and identifies the unit name to be used when allocating the local non-VSAM runtime data sets. KFJ_LOCAL_SMS_UNIT maps to the local value of RTE_SMS_UNIT that is used on the deployment target system.

This is a required field if the runtime data sets are not to be SMS-managed.

Required or optional
Required if the runtime data sets are not to be SMS-managed

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_SMS_VOLUME

Specifies the VOLSER for the local non-VSAM libraries.

This parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member, and identifies the volume serial number to be used when allocating the local non-VSAM runtime data sets. KFJ_LOCAL_SMS_VOLUME maps to the local value of RTE_SMS_VOLUME that is used on the deployment target system.

This is a required field if the runtime data sets are not to be SMS-managed.

Required or optional
Required if the runtime data sets are not to be SMS-managed

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_SMS_VSAM_MGMTCLAS

Specifies the MGMTCLAS for any local VSAM libraries.

This parameter specifies the SMS Management Class to be used when allocating the local VSAM runtime data sets. KFJ_LOCAL_SMS_VSAM_MGMTCLAS maps to the local value of RTE_SMS_VSAM_MGMTCLAS that is used on the deployment target system.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_SMS_VSAM_STORCLAS

Specifies the STORCLAS for local VSAM libraries.

The KFJ_LOCAL_SMS_VSAM_STORCLAS parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member, and identifies the SMS Storage Class to be used when allocating the local VSAM runtime data sets. KFJ_LOCAL_SMS_VSAM_STORCLAS maps to the local value of RTE_SMS_VSAM_STORCLAS that is used on the deployment target system.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_SMS_VSAM_VOLUME

Specifies the VOLSER for local VSAM libraries.

This parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member, and identifies the volume serial number to be used when allocating the local VSAM runtime data sets. KFJ_LOCAL_SMS_VSAM_VOLUME maps to the local value of RTE_SMS_VSAM_VOLUME that is used on the deployment target system.

This is a required field if the runtime data sets are not to be SMS-managed.

Required or optional
Required if the runtime data sets are not to be SMS-managed

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_TARGET_HILEV

Assigns the local SMP/E target high-level qualifier.

This parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member and identifies the high-level qualifier to be used when referencing the local SMP/E target data sets as part of the GENERATE action. The names of SMP/E target data sets will be generated by appending the appropriate suffix to this parameter.

KFJ_LOCAL_TARGET_HILEV maps to the local value of the GBL_TARGET_HILEV parameter, which is used on the deployment target system.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_USS_RTEDIR

Specify the local RTE HFS/zFS home directory.

This parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member. If any products in this RTE require z/OS UNIX System Services directories to be created, specify the local RTE HFS/zFS home directory. KFJ_LOCAL_USS_RTEDIR maps to the local value of RTE_USS_RTEDIR that is used on the deployment target system.

Required or optional
Required

Default value
`/var/rtehome`

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_LOCAL_USS_TKANJAR_PATH

Specifies the local z/OS UNIX System Services directory that the SMP/E installation jobs define using the ddname TKANJAR.

This parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$*lpar*) member. The **KFJ_LOCAL_USS_TKANJAR_PATH** parameter maps to the value of the **GBL_USS_TKANJAR_PATH** parameter that is used on the deployment target system.

Required or optional

Required if the runtime environment configures either of the following monitoring agents:

- CICS Transaction Gateway (TG). The corresponding configuration parameter is **CONFIGURE_CICS_TG_KGW**.
- Java Virtual Machine (JVM). The corresponding configuration parameter is **CONFIGURE_JVM_KJJ**.

Default value
/usr/lpp/kan/bin/IBM

Location where the parameter value is stored
This value is not stored in a configuration member.

Related parameters
"GBL_USS_TKANJAR_PATH" on page 73

KFJ_LOCAL_VSAM_HILEV

Specifies the local VSAM high-level qualifier.

This parameter is specified in the RTEDEF(PCK$PARM) or RTEDEF(PCK$lpar) member, and identifies the VSAM high-level qualifier to be used for allocating the local runtime data sets when the GENERATE action is run. The names of the VSAM data sets will be generated by appending the appropriate suffix to this parameter. KFJ_LOCAL_VSAM_HILEV maps to the local value of RTE_VSAM_HILEV that is used on the deployment target system.

Required or optional
Optional

Default value
%KFJ_LOCAL_PLIB_HILEV%

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_MIGRATE_WCONFIG

This parameter identifies the WCONFIG data set of the PARMGEN-configured runtime environment from which configuration settings are to be imported during migration.

Description

The **MIGRATE** action imports configuration settings from a runtime environment that is configured with PARMGEN to one that is configured with Configuration Manager. The **KFJ_MIGRATE_WCONFIG** parameter is used by the **MIGRATE** action to identify the WCONFIG data set of the source runtime environment (which is configured by PARMGEN).

Important: The source WCONFIG data set cannot be the same as the target WCONFIG data set. Make sure that the high-level qualifier specified for the source WCONFIG data set is not the same as the high-level qualifier of the target data sets (specified with the **RTE_PLIB_HILEV** parameter on the **MIGRATE** action).

Required or optional
Required for the **MIGRATE** action

Default value
None

Permissible values
highlevel.WCONFIG

Where *highlevel* is different from the target high-level qualifier specified in **RTE_PLIB_HILEV**

Example
TEST1.TST.DEMO.WCONFIG

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PACK_DATACLAS

Specifies the DATACLAS for non-VSAM package data sets.

This parameter is used by the PACKAGE and DEPLOY action, and specifies the SMS Data Class to be used when allocating or reading the non-VSAM package data sets. This parameter is not stored in any RTEDEF member and will be specified in the KCIVARS DD statement of your Configuration Manager batch JCL.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PACK_HILEV

Specifies the package non-VSAM high-level qualifier.

The KFJ_PACK_HILEV parameter is used by the PACKAGE and DEPLOY action, and specifies the high-level qualifier to be used when allocating or reading the non-VSAM package data sets. The names of the data sets to be allocated/read are generated by appending the appropriate suffix to this parameter.

This parameter is not stored in any RTEDEF member and will be specified in the KCIVARS DD statement of your Configuration Manager batch JCL.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PACK_MGMTCLAS

Specifies the MGMTCLAS for non-VSAM package data sets.

The KFJ_PACK_MGMTCLAS parameter is used by the PACKAGE and DEPLOY action, and specifies the SMS Management Class to be used when allocating or reading the non-VSAM package data sets. This parameter is not stored in any RTEDEF member and will be specified in the KCIVARS DD statement of your Configuration Manager batch JCL.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PACK_STORCLAS

Specifies the STORCLAS for non-VSAM package data sets.

The KFJ_PACK_STORCLAS parameter is used by the PACKAGE and DEPLOY action, and specifies the SMS Storage Class to be used when allocating or reading the non-VSAM package data sets. This parameter is not stored in any RTEDEF member and will be specified in the KCIVARS DD statement of your Configuration Manager batch JCL.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PACK_TERSE

Specifies to use the terse data set option.

The KFJ_PACK_TERSE parameter works as a switch during PACKAGE and DEPLOY actions:

- **Y** – Terse the packages created during PACKAGE action, that is, append suffix "TRS" to the package data set. Instruct the DEPLOY action to read tersed package data sets. Use KFJ_PACK_TERSE Y if you transfer package data sets using FTP.
- **N** – Do not terse the package data sets created by the PACKAGE action. The DEPLOY action will read the DUMP data sets using the documented naming convention. Use KFJ_PACK_TERSE N when working with Virtual Tapes.

This parameter is not stored in any RTEDEF member and will be specified in the KCIVARS DD statement of your Configuration Manager batch JCL.

Required or optional
Optional

Default value
N

Permissible values
Y, N

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PACK_UNIT

Specifies the Unit value for non-VSAM package data sets.

The KFJ_PACK_UNIT parameter is used by the PACKAGE and DEPLOY action, and specifies the UNIT name to be used when allocating or reading the non-VSAM package data sets. This parameter is not stored in any RTEDEF member and will be specified in the KCIVARS DD statement of your Configuration Manager batch JCL.

This parameter is required if the package data sets are not to be SMS-managed.

Required or optional
Required if the package data sets are not to be SMS-managed

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PACK_VOLUME

Specifies the volume for the non-VSAM package data set.

The KFJ_PACK_VOLUME parameter is used by the PACKAGE and DEPLOY actions, and specifies the volume serial number/name to be used when allocating or reading the non-VSAM package data sets. This parameter is not stored in any RTEDEF member and is specified in the KCIVARS DD statement of the Configuration Manager batch JCL.

This parameter is required if the package data sets are not to be SMS-managed.

Required or optional
Required if the package data sets are not to be SMS-managed

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PDCOL_HLQ

Specifies the high-level qualifier for the PDCOLLECT action.

The KFJ_PDCOL_HLQ parameter is used by the PDCOLLECT maintenance action in the TKANSAM(KFJMAINT) JCL job. This parameter is not stored in any RTEDEF member and is specified in the KCIVARS DD statement of the Configuration Manager batch JCL.

The parameter specifies the high-level qualifier for the TERSE data set that will be created as part of the PDCOLLECT action. The resulting data set will use the following pattern:

```
<kfj_pdcol_hlq>.PDCOLPDS.TRS
```

Required or optional
Optional

Default value
&SYSUID.KCIPDCOL

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PDCOL_JOB_ID

Specifies the Job ID of the started task.

Description

The KFJ_PDCOL_JOB_ID parameter is used by the PDCOLLECT maintenance action in the TKANSAM(KFJMAINT) JCL job. This parameter is not stored in any RTEDEF member and is specified in the KCIVARS DD statement of the Configuration Manager batch JCL.

The KFJ_PDCOL_JOB_ID parameter specifies the job ID located in the SDSF job output queue. It is used in combination with the KFJ_PDCOL_JOB_NAME parameter, and should point to the same job output.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PDCOL_JOB_NAME

Specifies the job name of the started task.

The KFJ_PDCOL_JOB_NAME parameter is used by the PDCOLLECT maintenance action only. The parameter is not stored in any RTEDEF member and is specified in the KCIVARS DD statement of the Configuration Manager batch JCL.

The parameter specifies the job name located in the SDSF job output queue.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_PDCOL_JOB_OUTPUT

Specifies the data set containing the job output of the started task.

Description

The KFJ_PDCOL_JOB_OUTPUT parameter is used by the PDCOLLECT maintenance action only. This parameter is not stored in any RTEDEF member and is specified in the KCIVARS DD statement of the Configuration Manager batch JCL.

The KFJ_PDCOL_JOB_OUTPUT parameter specifies the name of the data set that contains the job output of an OMEGAMON address space started task.

If the KFJ_PDCOL_JOB_NAME and KFJ_PDCOL_JOB_ID parameters are specified in the KCIVARS DD statement, the KFJ_PDCOL_JOB_OUTPUT parameter will be ignored.

Required or optional
Optional

Default value
None

Location where the parameter value is stored
This value is not stored in a configuration member.

KFJ_SECURITY_EXITS_LIB

This parameter specifies the name of the security exits library for the runtime environment.

Description

Use this parameter to specify the name of the data set that will contain the security exits for the runtime environment.

The default name for the security exits data set is *rte_plib_hilev*.*rte_name*.SECEXITS. By using the **KFJ_SECURITY_EXITS_LIB**, you can customize the name of the security exits data set.

The **KFJ_SECURITY_EXITS_LIB** parameter works with the **CREATE** and **MIGRATE** actions only.

If the specified data set does not exist when the **CREATE** or **MIGRATE** action runs, Configuration Manager allocates the data set and populates it with the default security exit members. For the **MIGRATE** action, this library will contain the security exit members from the source PARMGEN runtime environment.

Required or optional
Optional

Default value
rte_plib_hilev.*rte_name*.SECEXITS

Permissible values
A valid data set name

Example
TEST1.TST.DEMO.MYEXITS

Location where the parameter value is stored
This value is not stored in a configuration member.

Related parameters
"RTE_X_SECURITY_EXIT_LIB" on page 88

KFJ_SYSNAME

The **KFJ_SYSNAME** parameter is only needed if you use a system name (SYSNAME) that is larger than 4 characters (5 - 8 characters) **and** the SYSSMFID is not assigned or the default SYSSMFID setting is not acceptable.

Normally a SYSNAME is 1 - 4 characters in length. However, if a system has a longer SYSNAME, the **KFJ_SYSNAME** parameter can be used to override it as described below. The value listed for the **KFJ_SYSNAME** parameter will replace the value assigned to the **&SYSNAME** parameter when generating member names for the **DISCOVER**, **GENERATE**, and **MIGRATE** actions.

- If the value for **&SYSNAME** is 4 characters or less, the value for **KFJ_SYSNAME** will equal the **&SYSNAME**.
- If the value for **&SYSNAME** is 5 - 8 characters in length, the value for **KFJ_SYSNAME** will equal the **&SYSSMFID** parameter, which is the SMF ID.
- If the value for **&SYSSMFID** is not set explicitly, the system defaults to the CPU model number.

If the **KFJ_SYSNAME** value has been set for **&SYSSMFID**, warning message KFJ00205W will be issued in the **KCIPRINT** member. This message states that you can provide (override) a custom **KFJ_SYSNAME** value (in KCIVARS DD) when running the Monitoring Configuration Manager job. This allows you to avoid duplicate member names in the RTEDEF data set, in case the **&SYSSMFID** is set to default to the CPU model number.

You have to specify **KFJ_SYSNAME** if you are intending to create a runtime environment for a remote LPAR of the same or different SYSPLEX using a central configuration system. This will allow you to manage all your runtime environments from a central place and use the PACKAGE and DEPLOY actions to roll them out to the remote systems accordingly.

This parameter is specified in the KCIVARS DD statement of the JCL that executes the configuration manager Jobs (e.g TKANSAM(KFJJMCM).

Required?

This is only required if you must override the default value or use a central Configuration LPAR to pre-generate runtime environments for remote systems.

KFJ_USE_EMBEDS

This parameter controls whether override embed members are enabled for the RTE.

Description

A value of Y indicates that override embed members are enabled for the RTE.

For a **CREATE** or **MIGRATE** action, when this parameter is set to Y, Monitoring Configuration Manager creates a data set (the *embeds data set*) that contains the override embed parameters and values for the products that are installed in the respective CSI used to build the RTE. For the **MIGRATE** action, this library will contain the override embed parameters and values from the RTE being migrated from.

Unless specified otherwise in parameter **KFJ_EMBEDS_LIB**, the embeds data set name is *rte_plib_hilev*.*rte_name*.EMBEDS, where *rte_plib_hilev* is the high-level qualifier (HLQ) and *rte_name* is the name of the RTE specified on the action. By using this default naming convention, you can isolate the override embed values into the respective libraries per RTE.

Note: If the default or specified embeds data set exists when the **CREATE** or **MIGRATE** action runs, none of the existing members in it are changed or removed; only new members are added, which can occur if the CSI used to build the RTE has changed to add products.

During an initial **CREATE** or **MIGRATE** action when creating an RTE, when this parameter is set to Y, Monitoring Configuration Manager also defines the data set to the RTE using the **RTE_X_OVERRIDE_EMBEDS_LIB** parameter.

Required or optional

Optional

Default value
N

Permissible values
Y, N

Related parameters
KFJ_EMBEDS_LIB

RTE_X_OVERRIDE_EMBEDS_LIB

Target copy (TRG) parameters

The target copy parameters provide configuration settings for managing copies of SMP/E target libraries.

TRG_COPY_HILEV

Non-VSAM high-level qualifier for SMP/E target library copy

Description

This field specifies the high-level qualifier to be used when allocating the SMP/E target copy non-VSAM data sets. The names of these data sets will be generated by appending the appropriate suffix to this field.

Required or optional
Optional

Default value
%RTE_PLIB_HILEV%

TRG_COPY_MGMTCLAS

MGMTCLAS for non-VSAM libraries for SMP/E target library copy

Description

This field specifies the SMS management class to be used when allocating the SMP/E target copy non-VSAM data sets.

This field is required if the data sets are not to be SMS-managed. Leave this field blank if your installation does not use the SMS MGMTCLAS parameter or it is optional.

Required or optional
Optional

Default value
%RTE_SMS_MGTMCLAS%

TRG_COPY_NAME

SMP/E target library copy name

Description

This field specifies the name of the member in the RTEDEF library for SMP/E target copy settings. This member is required when using Configuration Manager to create and maintain an SMP/E target copy.

Required or optional
Optional

Default value
None

TRG_COPY_STORCLAS

STORCLAS for non-VSAM libraries for SMP/E target library copy

Description

This field specifies the SMS storage class to be used when allocating the SMP/E target copy non-VSAM data sets.

This field is required if the data sets are not to be SMS-managed. Leave this field blank if your installation does not use the SMS STORCLAS parameter or it is optional.

Required or optional
Optional

Default value
%RTE_SMS_STORCLAS%

TRG_COPY_TKANJAR_PATH

z/OS UNIX System Services directory for SMP/E target copy

Description

This field specifies the z/OS UNIX directory that will be used as a copy for the SMP/E installation jobs.

This field specifies the directory for a copy of SMP/E TKANJAR files, which is needed for OMEGAMON for CICS TG on z/OS (KGW) and OMEGAMON for JVM (KJJ).

TRG_COPY_UNIT

Unit for non-VSAM libraries for SMP/E target library copy

Description

This field specifies the unit name to be used when allocating the SMP/E target copy non-VSAM data sets.

This field is required if the data sets are not to be SMS-managed. Leave this field blank if your installation does not use the unit name or it is optional.

Required or optional
Optional

Default value
%RTE_SMS_UNIT%

TRG_COPY_VOLUME

VOLSER for non-VSAM libraries for SMP/E target library copy

Description

This field specifies the volume serial numbers to be used when allocating the SMP/E target copy non-VSAM data sets.

This field is required if the data sets are not to be SMS-managed. Leave this field blank if your installation does not use the volume serial number or it is optional.

Required or optional
Optional

Default value
%RTE_SMS_VOLUME%

Sparse parameter tables: The first row sets the default values for subsequent rows

For parameters that are arranged in tables, IBM Z Monitoring Configuration Manager uses the first row to set the defaults for subsequent rows.

You can use this behavior to specify "sparse" parameter tables. You only need to specify parameters whose values differ from the first row. Sparse parameter tables are especially useful for configuring the Db2 monitoring agent.

As with PARMGEN, you can also specify complete parameter tables: every parameter in every row.

Example: Db2 profile configuration

The following example shows a sparse parameter table that configures three Db2 profiles (PROD, TEST, and DEV):

```
KD2_PF                    BEGIN

KD2_PF01_ROW              01
KD2_PF01_PROFID           "PROD"
* Configure and autostart Near-Term History (NTH)
KD2_PF01_HIS_START        Y
* Store NTH data to VSAM data sets for e3270UI thread history
KD2_PF01_HIS_STORE        THVSAM
KD2_PF01_THRDHIS_LOG_NUM  10
* Storage units
KD2_PF01_HIS_VSAM_SU      CYLS
KD2_PF01_HIS_VSAM_MB      50

KD2_PF02_ROW              02
KD2_PF02_PROFID           "TEST"
* TEST requires fewer resources for NTH than PROD
KD2_PF02_THRDHIS_LOG_NUM  3
KD2_PF02_HIS_VSAM_MB      10

KD2_PF03_ROW              03
KD2_PF03_PROFID           "DEV"
* No NTH in DEV
KD2_PF03_HIS_START        N

KD2_PF                    END
```

The first row configures the PROD profile and sets the default values for subsequent rows (profiles).

The TEST profile omits the following parameters, falling back to the values from the first row:

KD2_PFnn_HIS_START
KD2_PFnn_HIS_STORE
KD2_PFnn_HIS_VSAM_SU

The TEST profile sets different, lower values than PROD for the following parameters, because TEST requires fewer resources for Near-Term History:

KD2_PF02_THRDHIS_LOG_NUM
KD2_PF02_HIS_VSAM_MB

The DEV profile does not configure Near-Term History.

Example: Db2 subsystem configuration

The following example shows a sparse parameter table that configures the Db2 monitoring agent to monitor three Db2 subsystems (DB2P, DB2T, and DB2D):

```
KD2_DB                    BEGIN

KD2_DB01_ROW              01
KD2_DB01_DB2_SSID         "DB2P"
KD2_DB01_DB2_DESCRIPTION  "PROD Db2 subsystem"
KD2_DB01_DB2_PROFID       "PROD"
```

```
KD2_DB01_DB2_VER           "12"
KD2_DB01_DB2_SYSNAME       "ZOSP"
KD2_DB01_DB2_DS_GROUP      "N"
KD2_DB01_DB2_MONITOR_START "Y"
KD2_DB01_DB2_PORT_NUM      "2000" * OMEGAMON Db2 PE Server TCP/IP port number
KD2_DB01_DB2_DSNTIAD       "DSNTIAD"

KD2_DB02_ROW               02
KD2_DB02_DB2_SSID          "DB2T"
KD2_DB02_DB2_DESCRIPTION   "TEST Db2 subsystem"
KD2_DB02_DB2_PROFID        "TEST"
KD2_DB02_DB2_SYSNAME       "ZOST"
KD2_DB02_DB2_PORT_NUM      "2001"

KD2_DB03_ROW               03
KD2_DB03_DB2_SSID          "DB2D"
KD2_DB03_DB2_DESCRIPTION   "DEVT Db2 subsystem"
KD2_DB03_DB2_PROFID        "DEVT"
KD2_DB03_DB2_SYSNAME       "ZOSD"
KD2_DB03_DB2_PORT_NUM      "2002"
KD2_DB03_DB2_LOADLIB       "DSN.VC10.SDSNLOAD"
KD2_DB03_DB2_RUNLIB        "DSN.VC10.RUNLIB.LOAD"

KD2_DB                     END
```

The first row configures the agent to monitor the DB2P subsystem and sets the default values for subsequent rows (subsystems).

The second row falls back to the values set by the first row for Db2 version, LOADLIB, and RUNLIB, but sets its own values for the profile, system (LPAR) name, and port number.

The third row specifies its own values for LOADLIB and RUNLIB.

Upgrade scenarios

This section describes how to add a new agent or product to an existing configuration RTEDEF library (that is, upgrade an existing RTEDEF) and mentions how to find more information on the Configuration Manager **MIGRATE** action to import existing legacy PARMGEN configuration settings.

Scenario: Configure new agent or product

Follow the steps below to configure a new agent or product:

1. Run the JCL with the **CREATE** action. The **RTE_NAME** and **RTE_PLIB_HILEV** parameters should point to an existing RTEDEF library (the one you want to modify). See "CREATE" on page 34 for more information.

   If you want to use override embed members, you should also specify Y for that parameter, for example, KFJ_USE_EMBEDS Y. See "Using override embed members" on page 133 for more information.

   The first step is to create new members in the RTEDEF library (for example, KDS$PARM), if necessary, where you can set product specific parameters. This step will also copy new override embed members and security exits (if there are any).

   Note: If the K*pp*$PARM member was not created for a new product, that means there are no mandatory or best practice parameters to set. However, if you want to add custom parameters, create the K*pp*$PARM member in the RTEDEF library and add custom parameters.

   When step 1 is complete, manually update the RTEDEF(*rte_name*) member and specify which agent or product to configure. For example, to configure the KD5 agent, set the **CONFIGURE_DB2_AGENT_KD5** parameter to Y.

   Note: If you want to remove an existing agent, set the parameter to N. The primary benefit of the N setting is that less CPU and time is used as the respective product-specific workflows are bypassed.
2. Manually update RTEDEF(GBL$PARM), if there are any initial mandatory parameters for the new product, such as load library names. See "Parameters in the initial runtime environment configuration profile" on page 69 for information on how to add those parameters to GBL$PARM and set correct values.
3. (Optional) Run JCL with the **DISCOVER** action. See "DISCOVER" on page 38 for more information.
4. Run the **GENERATE** action to finalize RTE configuration. See "GENERATE" on page 45 for more information.

Scenario: Upgrade to Configuration Manager from legacy PARMGEN

You can upgrade to Configuration Manager from an existing legacy PARMGEN installation using the **MIGRATE** action. For more information on **MIGRATE**, see "MIGRATE" on page 55.

Runtime environment definition (RTEDEF) library

A runtime environment definition is a set of parameters. Parameters are stored in a runtime environment definition (RTEDEF) library. The set of parameters for each runtime environment is organized into several RTEDEF members.

The data set name of the RTEDEF library consists of the high-level qualifiers that you specify to IBM Z Monitoring Configuration Manager in the **RTE_PLIB_HILEV** workflow variable in the KCIVARS input data set, followed by the fixed low-level qualifier RTEDEF:

rte_plib_hilev.RTEDEF

You can allocate the RTEDEF library yourself or you can use the **CREATE** action to allocate it and populate it with initial members for you.

An RTEDEF library can contain multiple runtime environment definitions. A single RTEDEF library can contain all of the runtime environment definitions for a sysplex. Or you can choose to store each runtime environment definition in a separate RTEDEF library.

If you are going to set up a High Availability TEMS (HA TEMS), make sure there is only one runtime environment defined in the RTEDEF (that is, the one used for the HA TEMS).

RTEDEF library members are also known as *configuration profile* members.

Related reference

Initial runtime environment library members
The **CREATE** action populates the RTEDEF library with an initial set of configuration profile members for a basic runtime environment.

Runtime environment definition library members

RTEDEF library members follow a naming convention that identifies the contents of the member and whether the member applies to a specific LPAR or to all LPARs.

Use the naming convention described in the following table to store parameters in the correct members. In the LPARs column of the following table, *Current* means the LPAR on which the **GENERATE** action is performed.

Table 9. RTEDEF member naming convention

Member name	Parameters	LPARs	Description
rte_name	**RTE_*** **CONFIGURE_***	All	Runtime environment configuration profile. *rte_name* matches the value of the **RTE_NAME** workflow variable in the KCIVARS data set of the job that performs the Monitoring Configuration Manager action.
KC5@*lpar* KD5@*lpar* KI5@*lpar* KN3@*lpar*	**Kpp_***	Current	LPAR-specific product configuration profile created by the **DISCOVER** action. **Note:** The **DISCOVER** action writes a comment member, K*pp*#*lpar*, if a K*pp*@*lpar* member already exists. For more information, see "Members created by the DISCOVER action" on page 41.

Table 9. RTEDEF member naming convention (continued)			
Member name	**Parameters**	**LPARs**	**Description**
SYS@*lpar*	Symbols	Current	LPAR-specific system symbols and KCIPARSE extracted variables found by the **DISCOVER** action. **Note:** The **DISCOVER** action writes a comment member, SYS#*lpar*, if a SYS@*lpar* member already exists. For more information, see "Members created by the DISCOVER action" on page 41.
K*pp*$PARM	**Kpp_***	All	Product configuration profile, where *pp* is the product code. For the list of supported code values, see "Products supported by Configuration Manager" on page 1.
K*pp*$*lpar*	**Kpp_***	Current	LPAR-specific product configuration profile, where *pp* is the product code. For the list of supported code values, see "Products supported by Configuration Manager" on page 1.
GBL$PARM	**GBL_***	All	Global configuration profile.
GBL$*lpar*	**GBL_***	Current	LPAR-specific global configuration profile.
VAR$GLOB	Variables	All	Variables configuration profile. For more information, see "Variables in parameter values" on page 129.
VAR$*lpar*	Variables	Current	LPAR-specific variables configuration profile. For more information, see "Variables in parameter values" on page 129.
PCK$PARM	**KFJ_LOCAL_***	All	Configuration profile created by the PACKAGE action that contains data set high-level qualifiers and other settings for deploying to a target system. For more information, see "Special considerations for SYSPLEX rollout" on page 111.
PCK$*lpar*	**KFJ_LOCAL_***	Current	LPAR-specific configuration profile created by the PACKAGE action that contains data set high-level qualifiers and other settings for deploying to a target system. For more information, see "Special considerations for SYSPLEX rollout" on page 111.
trg_copy_name	**TRG_COPY_*** **GBL_*** **CONFIGURE_***	All	Target copy configuration profile. *trg_copy_name* matches the value of the **TRG_COPY_NAME** workflow variable in the KCIVARS DD statement of the job that performs the Monitoring Configuration Manager action with option **TRGCOPY**.

The following figure illustrates an example of the RTEDEF library member naming and hierarchy. Depending on your configuration, there might be more or fewer members in your RTEDEF. The complete list of members is provided in the previous table.

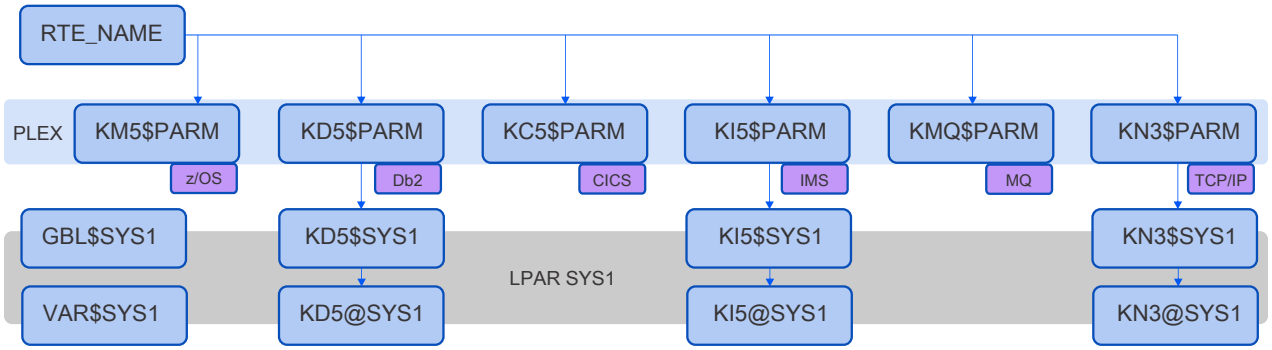

Figure 37. Example runtime environment definition library member naming and hierarchy

Note:

If a product has parameters with different prefixes, use the K*pp* from the corresponding **CONFIGURE_*** parameter as the member name prefix for all the parameters. For example:

- Store all Db2 agent parameters, **KD2_*** and **KD5_***, in KD5* members, to match the **CONFIGURE_DB2_AGENT_KD5** parameter.
- Store all **KC2_*** and **KC5_*** parameters in KC5* members.
- Store all **KI2_*** and **KI5_*** parameters in KI5* members.
- Store all **KDF_*** and **KS3_*** parameters in KS3* members.

Related tasks

Defining multiple runtime environments in an RTEDEF library
You can define one runtime environment per RTEDEF library or, as described here, you can define multiple runtime environments in a single RTEDEF library.

Concatenation order of runtime environment definition library members

The **GENERATE** action builds the set of parameters for a runtime environment by concatenating RTEDEF members in a well-defined order.

At a high level, the order in which the parameters are read can be categorized as follows:

- System symbols and KCIPARSE-extracted variables found by the **DISCOVER** action
- Target copy (if option **TRGCOPY** is specified)
- Variables (if system variables are enabled for the runtime environment)
- Runtime environment
- Product
- Global
- Package (if remote deployment applies)

The following table provides the specific order (by RTEDEF member) in which the parameters are read. In this list, parameters that are set in member SYS@*lpar* are read first and parameters that are set in member PCK$*lpar* are read last. In the following table, *lpar* identifies the LPAR on which the **GENERATE** action runs.

Important: If a parameter is set more than once, the last value that is read is used.

Order position	RTEDEF member	Notes
1	SYS@*lpar*	The SYS@*lpar* member is always processed first, even if the system variables flag (**RTE_SYSV_SYSVAR_FLAG**) is disabled.

Order position	RTEDEF member	Notes
2	*trg_copy_name*	The *trg_copy_name* member is processed only if option **TRGCOPY** is specified.
3	VAR$GLOB	The VAR$GLOB member is processed only if the system variables flag (**RTE_SYSV_SYSVAR_FLAG**) is enabled.
4	VAR$*lpar*	The VAR$*lpar* member is processed only if the system variables flag (**RTE_SYSV_SYSVAR_FLAG**) is enabled.
5	*rte_name*	
6	K*pp*@*lpar*	For KC5, KI5, KD5, and KN3 only
7	K*pp*$PARM	
8	K*pp*$*lpar*	
9	GBL$PARM	
10	GBL$*lpar*	
11	PCK$PARM	The PCK$PARM member is processed only if the runtime environment is designed for remote deployment (that is, with **KFJ_LOCAL_PLIB_HILEV** specified in the KCIVARS DD statement).
12	PCK$*lpar*	The PCK$*lpar* member is processed only if the runtime environment is designed for remote deployment (that is, with **KFJ_LOCAL_PLIB_HILEV** specified in the KCIVARS DD statement).

Tip: After you run the **GENERATE** action, you can view an ordered list of the RTEDEF library members that the action uses. For more information, see "Parameter values used" on page 149.

Initial runtime environment library members

The **CREATE** action populates the RTEDEF library with an initial set of configuration profile members for a basic runtime environment.

There are hundreds of OMEGAMON parameters. However, to configure a runtime environment that uses basic functions, you only need to specify the few dozen parameters in these initial members. All other parameters use their default values.

Edit the members to specify appropriate parameters for your runtime environment.

Characteristics of the runtime environment defined by the initial members

The parameters in the initial members define a runtime environment with the following characteristics:

Table 10. Characteristics of the initial runtime environment definition	
Characteristic	**Defined by these parameters**
Static hub monitoring server.	`KDS_TEMS_TYPE HUB` Default parameter value (not specified in the initial set of parameters): `KDS_TEMS_HA_TYPE` *none*
All *installed* components configured. The **CREATE** action detects which component products, such as monitoring agents, are installed and sets the corresponding **CONFIGURE_*** parameters to Y.	`CONFIGURE_* Y`

Runtime members shared with target libraries.
Rather than being a full stand-alone set, the runtime members refer to some SMP/E installation target libraries (or, more typically, a copy that you have created of those target libraries).
	Default parameter values (not specified in the initial set of parameters):
	`“RTE_TYPE” on page 79 SHARING`
	`RTE_SHARE SMP`

The initial members include only a few product-specific members

Only a few component products require you to specify parameter values for their basic functions. Most work out-of-the-box using default parameter values.

The **CREATE** action creates K*pp*$PARM members only for the following components, and only if they are installed:

Monitoring server (KDS$PARM)
CICS TG monitoring agent (KGW$PARM)

The **CREATE** action creates these members based only on which components are installed. The **CREATE** action is not sensitive to **CONFIGURE_*** parameters in an existing RTEDEF(*rte_name*) member. The **CREATE** action neither reads nor overwrites existing RTEDEF members. For example, if the CICS TG monitoring agent is installed in the target libraries, then the **CREATE** action creates an RTEDEF(KGW$PARM) member, even if an RTEDEF(*rte_name*) member already exists and specifies **CONFIGURE_CICS_TG_KGW** N.

To set parameter values for other products, you must create the corresponding product-specific K*pp** members.

Special considerations for SYSPLEX rollout

Specify an additional parameter, **KFJ_LOCAL_PLIB_HILEV**, in your KCIVARS DD statement when running the **CREATE** action to have the RTEDEF created using the value in **KFJ_LOCAL_PLIB_HILEV**.

When this parameter is specified, the generated KFJ_LOCAL_PLIB_HILEV.RTEDEF data set will contain an additional member called PCK$PARM. This member allows locally generated RTE data sets to have a different HLQ than the data sets that will be used on the deployment (target) system.

Just like for other members in the RTEDEF, there is a SYSPLEX-wide member called RTEDEF(PCK$PARM) and an LPAR-specific member, RTEDEF(PCK$lpar), supported. The **CREATE** action will generate the RTEDEF(PCK$PARM) member.

If any of the members in PCK$PARM are not defined, Configuration Manager will use the **KFJ_LOCAL_PLIB_HILEV** parameter to generate default values for all of the high-level qualifier parameters. For the remaining parameters, the target RTE parameter will be used as the default.

The target RTE parameter will be used to allocate the respective data sets.

For more information about the PCK$PARM member, see “RTEDEF(PCK$PARM)” on page 114.

Using an SMP/E target copy

Specifying the **CREATE** action with option **TRGCOPY** creates member RTEDEF(*trg_copy_name*). This member contains source and destination information used to create and maintain a copy of your SMP/E target libraries. This member is used only when the **CREATE** or **GENERATE** action is run using option **TRGCOPY**. For more information, see “RTEDEF(trg_copy_name)” on page 116 and “Using SMP/E target library copies” on page 141.

Related tasks

CREATE
The **CREATE** action creates an initial runtime environment definition that you can customize to match your requirements.

Related reference

Members created by the DISCOVER action
The **DISCOVER** action creates members for each type of subsystem it discovers.

Runtime environment definition (RTEDEF) library
A runtime environment definition is a set of parameters. Parameters are stored in a runtime environment definition (RTEDEF) library. The set of parameters for each runtime environment is organized into several RTEDEF members.

RTEDEF(*rte_name*)

The RTEDEF(*rte_name*) member is the runtime environment configuration profile. This member contains parameters with the prefixes **RTE** and **CONFIGURE**.

```
RTE_NAME                          <rte_name>
RTE_PLIB_HILEV                    <rte_plib_hilev>
RTE_SECURITY_USER_LOGON           NONE
RTE_SECURITY_FOLD_PASSWORD_FLAG   Y
RTE_SECURITY_CLASS                ""
RTE_X_SECURITY_EXIT_LIB           <rte_plib_hilev>.<rte_name>.SECEXITS
RTE_TEMS_NAME_NODEID              <rte_name>:TEMS:RTE_COMM_PROTOCOL1
RTE_COMM_PROTOCOLn               IPPIPE
RTE_TCP_HOST                      <sysiphostname>
RTE_TCP_PORT_NUM                  1918
RTE_VTAM_APPLID_PREFIX            OM<sysclone>
RTE_STC_PREFIX                    OMEG
RTE_USS_RTEDIR                    "/var/rtehome"
CONFIGURE_TEMS_KDS                Y  * TEMS
CONFIGURE_E3270UI_KOB             Y  * Enhanced 3270
CONFIGURE_CICS_KC5                Y  * CICS TS
CONFIGURE_CICS_TG_KGW             Y  * CICS TG
CONFIGURE_DB2_AGENT_KD5           Y  * Db2
CONFIGURE_IMS_KI5                 Y  * IMS
CONFIGURE_JVM_KJJ                 Y  * JVM
CONFIGURE_ZOS_KM5                 Y  * z/OS
CONFIGURE_MESSAGING_KMQ           Y  * MQ
CONFIGURE_MESSAGING_KQI           Y  * Integration Bus
CONFIGURE_NETVIEW_KNA             Y  * Netview
CONFIGURE_MFN_KN3                 Y  * Network
CONFIGURE_STORAGE_KS3             Y  * Storage
CONFIGURE_OMEGAVIEW_KWO           Y  * Integration Monitor
CONFIGURE_ITCAMAD_KYN             Y  * ITCAM for Applications
CONFIGURE_ACM_KRN                 Y  * Advanced Catalog Mgmt
CONFIGURE_ARD_KRH                 Y  * Advanced Rpt and Mgmt
CONFIGURE_AAD_KRG                 Y  * Advanced Audit
CONFIGURE_AAM_KRJ                 Y  * Advanced Alloc Mgmt
CONFIGURE_ATAM_KRK                Y  * Automated Tape Alloc
CONFIGURE_ABR_KRV                 Y  * Advanced Backup and Rec
```

*Figure 38. Initial RTEDEF(rte_name) member created by the **CREATE** action*

Note: RTE_X_OVERRIDE_EMBEDS_LIB is included if you specify **KFJ_USE_EMBEDS** set to Y.

Non-default values

The values of the following parameters in this initial member are different than the default values:

RTE_TEMS_NAME_NODEID
This parameter sets the node ID of the monitoring server that is configured in this runtime environment. The only difference between the value in this initial member and the default value: the default value ends in CMS rather than TEMS. Given the related parameter names, TEMS is a more intuitive value.

RTE_VTAM_APPLID_PREFIX
LPARs in a sysplex might have their own instances of the same VTAM application. The fixed default VTAM applid prefix, CTD, does not help to identify the LPAR to which each instance belongs.

This initial member sets the value to OM<*sysclone*>, where:

- OM is an abbreviation of OMEGAMON.
- <*sysclone*> is the 1- or 2-character value of the z/OS static system symbol **SYSCLONE**. **SYSCLONE** is shorthand notation for the system (LPAR) name.

RTE_STC_PREFIX
The default started task prefix is IBM.

This initial member sets the value to OMEG (an abbreviation of OMEGAMON).

Significant default values

Some characteristics of the runtime environment configured using this initial member are determined by the following default parameter values:

```
RTE_TYPE                        SHARING
RTE_SHARE                       SMP
```

RTEDEF(KDS$PARM)

The RTEDEF(KDS$PARM) member contains parameters that configure the monitoring server. These parameters have the prefix **KDS**.

```
* Tivoli Enterprise Monitoring Server

KDS_TEMS_TYPE     HUB
```

*Figure 39. Initial RTEDEF(KDS$PARM) member created by the **CREATE** action*

RTEDEF(KGW$PARM)

The RTEDEF(KDS$PARM) member contains parameters that configure the CICS Transaction Gateway monitoring agent. These parameters have the prefix **KGW**.

```
* CICS Transaction Gateway

KGW_SA                          BEGIN               * Table begin *
KGW_SA01_ROW                    01
KGW_SA01_CTG_DAEMON_STC         CTGPROC   * Sample CTG Daemon *
KGW_SA01_CTG_DAEMON_PORT_NUM    2980      * 00000-65535
KGW_SA01_CTG_DAEMON_HOST        LOCALHOST
KGW_SA01_SAPI_CLIENT_CTGTRACE   0       * 0-4
* END KGW_SA01 row 1 (add more rows as needed!)
KGW_SA                          END                 * Table end   *
```

*Figure 40. Initial RTEDEF(KGW$PARM) member created by the **CREATE** action*

RTEDEF(GBL$PARM)

The RTEDEF(GBL$PARM) member is the global configuration profile. This member contains global parameters. These parameters have the prefix **GBL**.

The **CREATE** action only populates the RTEDEF(GBL$PARM) member with parameters that are relevant to the monitoring agents that are installed. For example:

```
* Global parameters (used by installed products)

* High-level qualifier of SMP/E target libraries
GBL_TARGET_HILEV        "<HLQs of KCIFLOW dsname of CREATE action>"

* Java home directory (KYN, KDS)
GBL_HFS_JAVA_DIR1       "/usr/lpp/java/IBM/J8.0_64"

* SMP/E target directory containing TKANJAR files (KGW, KJJ)
GBL_USS_TKANJAR_PATH    "/usr/lpp/kan/bin/IBM"

* ICSF load library containing CSNB* modules for password encryption
* (KS3, KI5)
GBL_DSN_CSF_SCSFMOD0    "CSF.SCSFMOD0"

* Db2 libraries (KD5)
GBL_DSN_DB2_RUNLIB_V12  "DSN.VCR1M0.RUNLIB.LOAD"
GBL_DSN_DB2_LOADLIB_V12 "DSN.VCR1M0.SDSNLOAD"
GBL_DSN_DB2_DSNEXIT     "DSN.VCR1M0.DSNEXIT"

* CICS Transaction Gateway (KGW)
GBL_DSN_CICS_CTG_DLL    "SYS1.SCTGDLL"

* IMS libraries (KI5)
GBL_DSN_IMS_RESLIB      "IMS.SDFSRESL"
GBL_DSN_IMS_SCEXLINK    "IMS.SCEXLINK"
GBL_DSN_IMS_SFUNLINK    "IMS.SFUNLINK"

* MQ and Integration Broker (KMQ, KQI)
GBL_DSN_WMQ_SCSQANLE    "CSQ.V9R0M0.SCSQANLE"
GBL_DSN_WMQ_SCSQAUTH    "CSQ.V9R0M0.SCSQAUTH"

* Netview CNMLINK library
GBL_DSN_NETVIEW_CNMLINK "NETVIEW.VNRNMN.CNMLINK"
```

*Figure 41. Initial RTEDEF(GBL$PARM) member created by the **CREATE** action*

GBL_TARGET_HILEV
The **CREATE** action sets the **GBL_TARGET_HILEV** parameter to the high-level qualifiers of the data set name specified by the KCIFLOW DD statement in the job step that performs the **CREATE** action.

Combined with the default parameter values **RTE_TYPE** SHARING and **RTE_SHARE** SMP, this value configures the runtime environment to share the same target library that was used to perform the **CREATE** action.

RTEDEF(PCK$PARM)

The RTEDEF(PCK$PARM) member contains parameters that allow remote deployment of a runtime environment.

The RTEDEF(PCK$PARM) member contains the values described in the following table:

PCK$PARM or PCK$lpar member name	Target RTE parameter mapped in RTEDEF(rte_name)	Comment/description
KFJ_LOCAL_PLIB_HILEV	RTE_PLIB_HILEV	
KFJ_LOCAL_HILEV	RTE_HILEV	
KFJ_LOCAL_VSAM_HILEV	RTE_VSAM_HILEV	Default VSAM HLQ to be used when allocating VSAM data sets

PCK$PARM or PCK$lpar member name	Target RTE parameter mapped in RTEDEF(rte_name)	Comment/description
KFJ_LOCAL_PDS_HILEV	RTE_PDS_HILEV	Default PDS HLQ to be used to allocate PDS V1 data sets (if needed)
KFJ_LOCAL_TARGET_HILEV	GBL_TARGET_HILEV	SMP/ Target HLQ found in RTEDEF(GBL$PARM/ GBL$lpar)
KFJ_LOCAL_USS_RTEDIR	RTE_USS_RTEDIR	RTE z/OS UNIX System Services root path used by Configuration Manager to save all z/OS UNIX-related artifacts
KFJ_LOCAL_USS_TKANJAR_PATH	GBL_USS_TKANJAR_PATH	SMP/E target directory containing TKANJAR files
KFJ_LOCAL_SMS_VOLUME	RTE_SMS_VOLUME	To support non-SMS environments
KFJ_LOCAL_SMS_VSAM_VOLUME	RTE_SMS_VSAM_STORCLAS	To support non-SMS environments
KFJ_LOCAL_SMS_UNIT	RTE_SMS_UNIT	To support non-SMS environments
KFJ_LOCAL_SMS_STORCLAS	RTE_SMS_STORCLAS	
KFJ_LOCAL_SMS_VSAM_STORCLAS	RTE_SMS_VSAM_STORCLAS	
KFJ_LOCAL_SMS_MGMTCLAS	RTE_SMS_MGMTCLAS	
KFJ_LOCAL_SMS_VSAM_MGMTCLAS	RTE_SMS_VSAM_MGMTCLAS	

Sample member RTEDEF(PCK$PARM)

The sample member, RTEDEF(PCK$PARM), is shown below.

```
* Local RTE parameters

KFJ_LOCAL_PLIB_HILEV          TSOUID.LOCAL
KFJ_LOCAL_HILEV
KFJ_LOCAL_VSAM_HILEV
KFJ_LOCAL_PDS_HILEV

* High-level qualifier of local SMP/E target libraries
KFJ_LOCAL_TARGET_HILEV        "MONSUITE"

* Path to local USS files
KFJ_LOCAL_USS_RTEDIR          "/var/rtehome"

* Local SMP/E target directory containing TKANJAR files (KGW, KJJ)
KFJ_LOCAL_USS_TKANJAR_PATH    "/usr/lpp/kan/bin/IBM"

* SMS Parameters
KFJ_LOCAL_SMS_VOLUME
KFJ_LOCAL_SMS_VSAM_VOLUME
KFJ_LOCAL_SMS_UNIT
KFJ_LOCAL_SMS_STORCLAS
KFJ_LOCAL_SMS_VSAM_STORCLAS
KFJ_LOCAL_SMS_MGMTCLAS
KFJ_LOCAL_SMS_VSAM_MGMTCLAS
```

Figure 42. Example of PCK$PARM

This member essentially specifies a mapping. A shadow value is specified that represents the value used to generate the runtime environment data sets on the local system (that is, the configuration LPAR). Note that this is the only place this mapping is done. All other values in the RTEDEF members should be specified using the attributes and HLQs of the target system (that is, the system that the RTE will be deployed to).

The list of **HILEV** parameters, along with **RTE_USS_RTEDIR** and **GBL_USS_TKANJAR_PATH**, are the only parameters supported for this mapping. If the **GENERATE** action detects that non-supported HLQs are being used in the RTEDEF, an error message will be displayed in KCIPRINT indicating incompatible parameters, and the **GENERATE** action workflow stops.

Example:

The **RTE_PLIB_HILEV** value in RTEDEF(MYRTE) is as follows:

```
RTE_PLIB_HILEV                  TDCIT.REG
```

Specifying **KFJ_LOCAL_PLIB_HILEV** with a value of SYS3.PREGEN in RTEDEF(PCK$PARM) will allocate the data sets that will use a value of **RTE_PLIB_HILEV** using SYS3.PREGEN instead of TDCIT.REG. When using the **DEPLOY** action after having run the **PACKAGE** action and transferring the dump data sets to the target system, the original value as specified in RTEDEF(MYRTE), that is TDCIT.REG in this specific example, will be used to restore the data sets.

RTEDEF(*trg_copy_name*)

The RTEDEF(*trg_copy_name*) member is the SMP/E target copy configuration profile. This member contains parameters with the prefixes **GBL**, **CONFIGURE**, and **TRG**.

```
* High-level qualifier of SMP/E target libraries
GBL_TARGET_HILEV                "<HLQs of KCIFLOW dsname of CREATE action>"

* SMP/E target directory containing TKANJAR files (KGW, KJJ)
GBL_USS_TKANJAR_PATH            "/usr/lpp/kan/bin/IBM"

TRG_COPY_NAME                   <trg_copy_name>
* High-level qualifier of the copy of SMP/E target libraries
TRG_COPY_HILEV                  <trg_copy_hilev>
* Directory for a copy of SMP/E TKANJAR files (KGW, KJJ)
TRG_COPY_TKANJAR_PATH           "/var/rtehome/<trg_copy_name>/kan/bin/IBM"

CONFIGURE_TEMS_KDS              Y  * TEMS
CONFIGURE_E3270UI_KOB           Y  * Enhanced 3270
CONFIGURE_CICS_KC5              Y  * CICS TS
CONFIGURE_CICS_TG_KGW           Y  * CICS TG
CONFIGURE_DB2_AGENT_KD5         Y  * Db2
CONFIGURE_IMS_KI5               Y  * IMS
CONFIGURE_JVM_KJJ               Y  * JVM
CONFIGURE_ZOS_KM5               Y  * z/OS
CONFIGURE_MESSAGING_KMQ         Y  * MQ
CONFIGURE_MESSAGING_KQI         Y  * Integration Bus
CONFIGURE_NETVIEW_KNA           Y  * Netview
CONFIGURE_MFN_KN3               Y  * Network
CONFIGURE_STORAGE_KS3           Y  * Storage
CONFIGURE_OMEGAVIEW_KWO         Y  * Integration Monitor
CONFIGURE_ITCAMAD_KYN           Y  * ITCAM for Applications
CONFIGURE_ACM_KRN               Y  * Advanced Catalog Mgmt
CONFIGURE_ARD_KRH               Y  * Advanced Rpt and Mgmt
CONFIGURE_AAD_KRG               Y  * Advanced Audit
CONFIGURE_AAM_KRJ               Y  * Advanced Alloc Mgmt
CONFIGURE_ATAM_KRK              Y  * Automated Tape Alloc
CONFIGURE_ABR_KRV               Y  * Advanced Backup and Rec
```

*Figure 43. Initial RTEDEF(trg_copy_name) member created by the **CREATE** action with option **TRGCOPY***

GBL_*
The source location from where a copy is taken

TRG_COPY_*
The target destination where the files are copied to

CONFIGURE_*
Products for which configuration files will be copied from the source (**GBL_***) to the destination (**TRG_COPY_***)

Additional TRG_* parameters

Target copy libraries are always non-VSAM. You can also include the following parameters in your SMP/E target copy member when allocating SMP/E target copy libraries:

TRG_COPY_VOLUME
Specifies the VOLSER for target copy non-VSAM libraries.

TRG_COPY_UNIT
Specifies the unit name for target copy non-VSAM libraries.

TRG_COPY_STORCLAS
Specifies the STORCLAS for target copy non-VSAM libraries.

TRG_COPY_MGMTCLAS
Specifies the MGMTCLAS for target copy non-VSAM libraries.

Runtime members

The **GENERATE** action generates runtime members in locations (MVS data set names and z/OS UNIX System Services paths) that are specified by parameters.

Runtime member locations

MVS data sets:

rte_hilev.*rte_name*.RK*
rte_hilev.*rte_name*.*ssid*.RK*
rte_vsam_hilev.*rte_name*.RK*
rte_vsam_hilev.*rte_name*.*ssid*.RK*
gbl_dsn_sys1_proclib
gbl_dsn_sys1_vtamlib
gbl_dsn_sys1_vtamlst

z/OS UNIX directory paths:

rte_uss_dir/*rte_name*/*

where:

- *ssid* is the identifier of a subsystem to be monitored, such as a Db2 subsystem or an IMS subsystem
- Other identifiers in *italics* represent parameter values

Parameters that specify runtime member locations

RTE_HILEV
High-level qualifiers of non-VSAM data sets.

RTE_VSAM_HILEV
High-level qualifiers of VSAM data sets.

RTE_HILEV and **RTE_VSAM_HILEV** have the same default value: *rte_plib_hilev*

GBL_DSN_SYS1_PROCLIB
GBL_DSN_SYS1_VTAMLIB
GBL_DSN_SYS1_VTAMLST
The data set names where the **GENERATE** action writes members intended for your system libraries.

Default values:

rte_hilev.SYS1.PROCLIB
rte_hilev.SYS1.VTAMLIB
rte_hilev.SYS1.VTAMLST

You must use your own site-specific procedures to copy the members from these locations to your actual PROCLIB, VTAMLIB, and VTAMLST system libraries.

RTE_USS_RTEDIR
z/OS UNIX directory. Default value: /var/rtehome

RTE_NAME
The runtime environment name is appended to **RTE_HILEV** and **RTE_VSAM_HILEV** as a low-level qualifier and to **RTE_USS_RTEDIR** as a directory name.

The following diagram illustrates the relationships between parameters and runtime member locations:

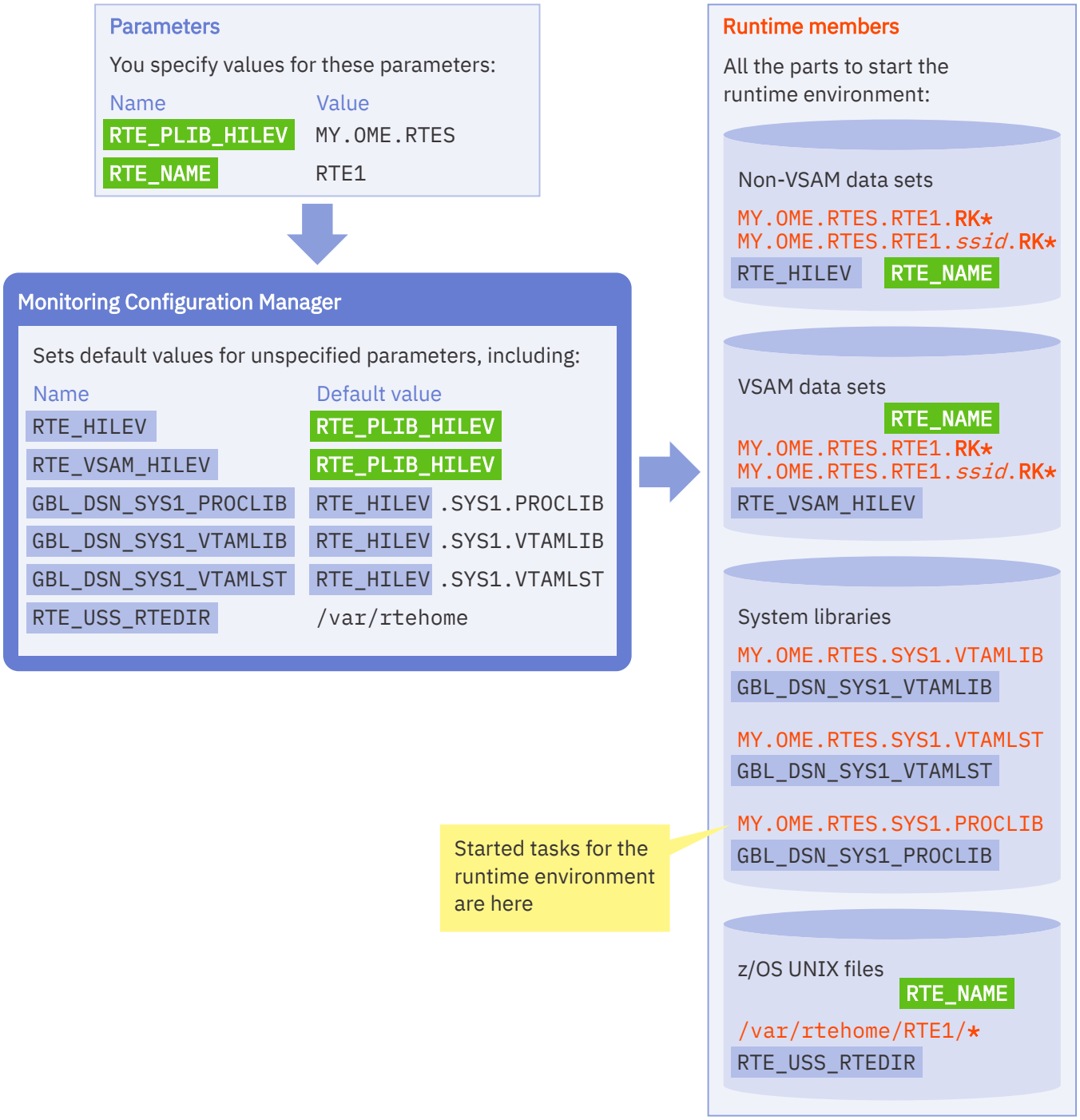

Figure 44. How parameters affect the locations of runtime members

Tip: Monitoring Configuration Manager writes concise started tasks to:

rte_hilev.SYS1.PROCLIB

and versions with verbose comments to the same location used by PARMGEN:

rte_plib_hilev.*rte_name*.RKANSAMU
rte_plib_hilev.*rte_name*.RKD2SAM (for Db2)

Related tasks
GENERATE

The **GENERATE** action generates runtime members for a runtime environment from a set of configured parameters.

Related reference

RTE_NAME
The runtime environment name.

RTE_PLIB_HILEV
The default high-level qualifiers of runtime members that are stored in MVS data sets.

RTE_USS_RTEDIR
The path where runtime members are stored in z/OS UNIX System Services.

Communication between monitoring components

In a typical topology, monitoring agents communicate with remote monitoring servers, and remote monitoring servers communicate with a single, central hub monitoring server.

Typical topology

The following diagram shows a simple example.

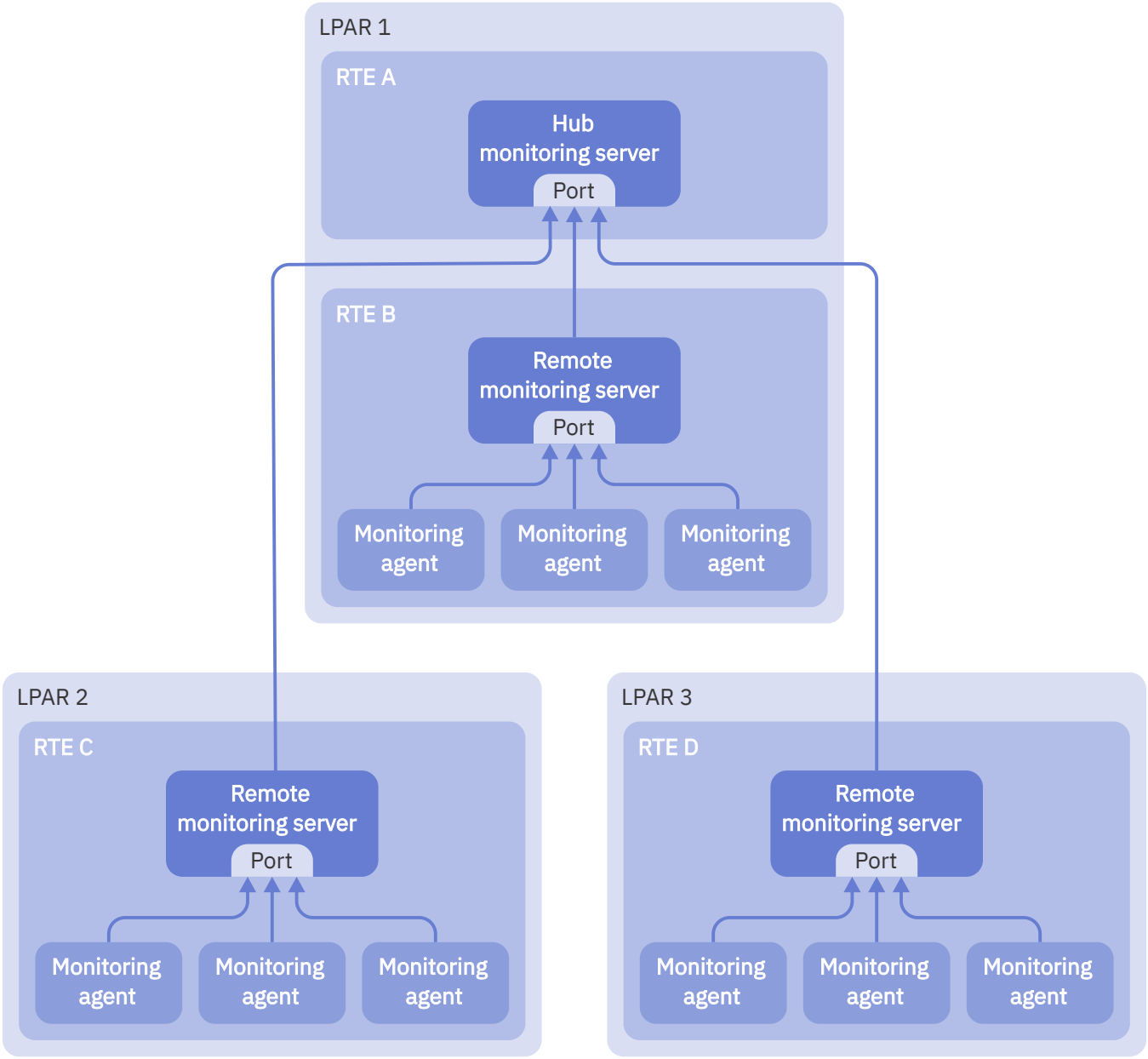

Figure 45. Typical topology of runtime environments in a sysplex

The number of monitoring agents in a runtime environment and the number of LPARs depends on your site.

In a typical topology, monitoring agents communicate with the remote monitoring server that is in the same runtime environment as the agents. Remote monitoring servers are described as *remote* to distinguish them from the *hub* monitoring server. Remote monitoring servers are typically *local* to

the monitoring agents with which they communicate, in the sense that they are in the same runtime environment and, hence, the same LPAR.

The default communication protocol for all components is Transport Control Protocol over Internet Protocol version 4 (TCP/IPv4). The hub monitoring server listens on a port for messages from remote monitoring servers. Remote monitoring servers listen on a port for messages from monitoring agents.

Required parameters

Communication between components involves several parameters. However, for most of these parameters you can use default values.

To configure communication between a remote monitoring server and a hub server, you only need to specify the following parameters:

- In the runtime environment that contains the hub monitoring server (in the previous figure, RTE A):

  KDS_TEMS_TYPE
  HUB, rather than the default REMOTE.

  RTE_TCP_PORT_NUM
  The port on which the hub listens for messages from remote monitoring servers.

- In a runtime environment that contains a remote monitoring server (such as RTE D):

  KDS_HUB_TEMS_NAME_NODEID
  Must match the **RTE_TEMS_NAME_NODEID** parameter of the hub. The default value of **RTE_TEMS_NAME_NODEID** is *rte_name*:CMS.

  KDS_TCP_PORT_NUM
  Must match the **RTE_TCP_PORT_NUM** parameter of the hub.

  KDS_HUB_TCP_HOST
  Must match the host name or IP address of the LPAR that contains the hub.

You don't need to specify any parameters to configure communication between monitoring agents and a remote monitoring server in the same runtime environment.

The following diagram shows the required parameters in each runtime environment and their relationships:

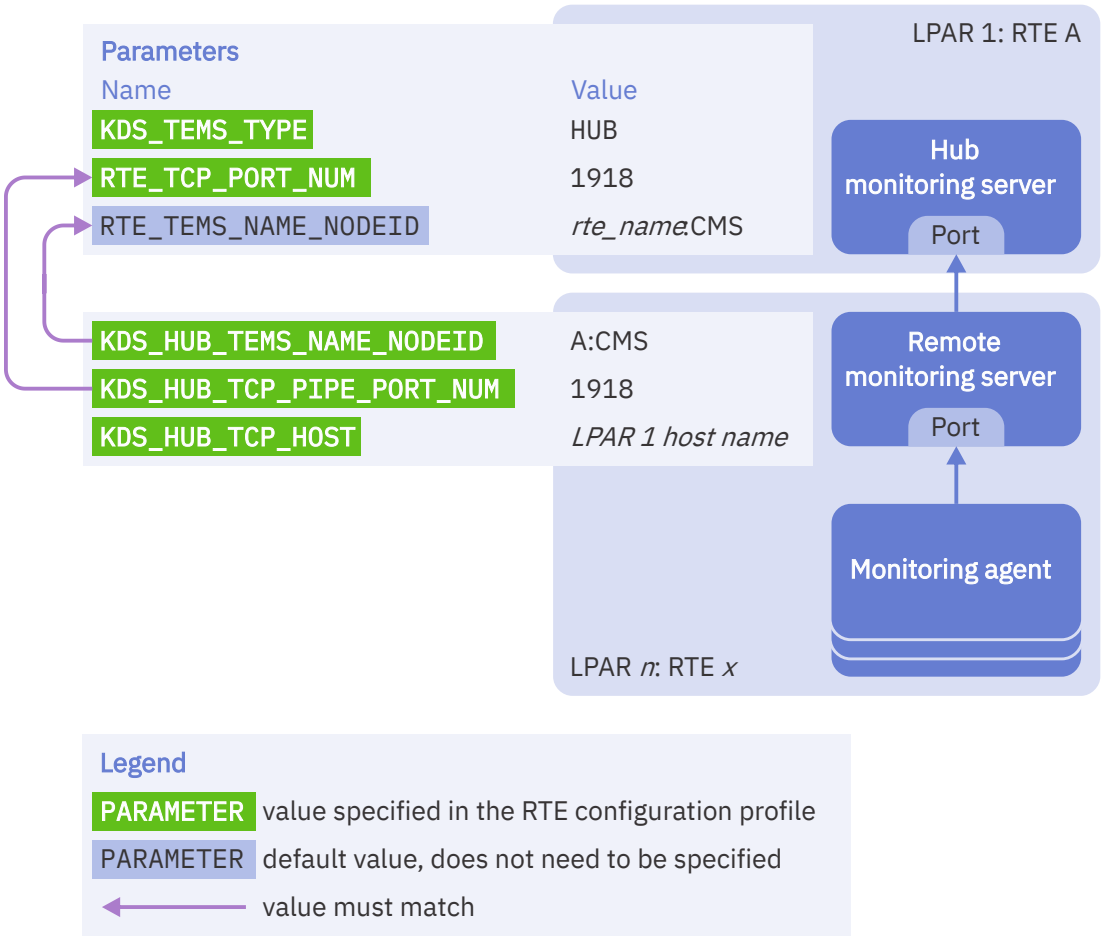

Figure 46. Parameters required to configure a typical topology

Other parameters

The following diagram shows a more comprehensive overview of the parameters to configure communication between components, including the default values of parameters omitted from the previous diagram.

For a typical topology, you do not need to be aware of these other parameters. This diagram is provided as a reference for configuring different topologies.

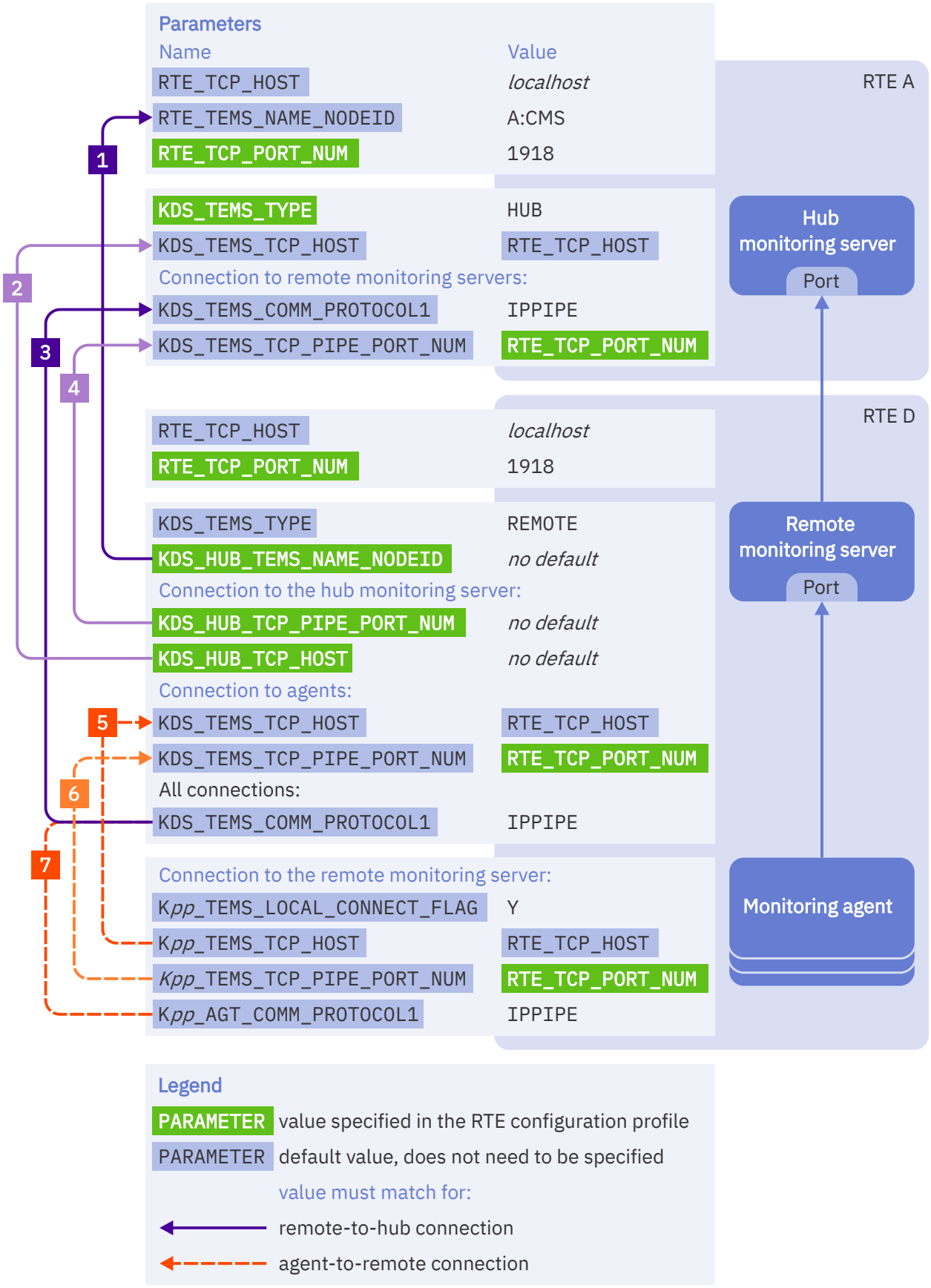

Figure 47. Parameters to configure communication between components, including significant default values

Communication between a remote monitoring server and the hub monitoring server

For a remote monitoring server to communicate with the hub monitoring server, the following parameters must match:

n figure label	This parameter in the RTE of the remote monitoring server	...must match this parameter in the RTE of the hub
1	**KDS_HUB_TEMS_NAME_NODEID** The remote monitoring server must refer to the node ID of the hub.	**RTE_TEMS_NAME_NODEID**
2	**KDS_HUB_TCP_HOST** The remote monitoring server must refer to the host name of the hub.	**KDS_TEMS_TCP_HOST**
3	**KDS_TEMS_COMM_PROTOCOLn** The remote monitoring server and hub must have at least one communication protocol in common.	**KDS_TEMS_COMM_PROTOCOLn**
4	**KDS_HUB_TCP_PIPE_PORT_NUM** The remote monitoring server must refer to the port on which the hub is listening.	**KDS_TEMS_TCP_PIPE_PORT_NUM**

Communication between monitoring agents and a remote monitoring server

For monitoring agents to communicate with a remote monitoring server, the following parameters must match:

n figure label	This parameter for the monitoring agent...	...must match this parameter for the remote monitoring server
5	**Kpp_TEMS_TCP_HOST**	**KDS_TEMS_TCP_HOST**
6	**Kpp_TEMS_TCP_PIPE_PORT_NUM**	**KDS_TEMS_TCP_PIPE_PORT_NUM**
7	**Kpp_AGT_COMM_PROTOCOLn**	**KDS_TEMS_COMM_PROTOCOLn**

If monitoring agents communicate with the remote monitoring server that is in the same runtime environment, as specified by the default parameter value **Kpp_TEMS_LOCAL_CONNECT_FLAG** Y, then all of these parameters match by default.

Choice of communication protocols

You can specify up to seven communication protocols for each monitoring agent and server. When attempting to communicate, a monitoring agent or server tries its protocols in order. If the first choice fails, it tries the second choice, and so on. You can either set the communication protocol choices individually for the monitoring server and each agent in a runtime environment, or you can use the **RTE_COMM_PROTOCOLn** parameters to set them all together.

Related tasks

Converting a hub monitoring server to a remote monitoring server

Initially, you might configure a new runtime environment to be stand-alone, with its own hub monitoring server. Later, you can integrate that runtime environment with the rest of your monitoring topology by converting its hub monitoring server to a remote monitoring server that communicates with a central hub.

Related reference

RTE_COMM_PROTOCOLn
Sets the communication protocol choices of all components in the runtime environment.

Variables in parameter values

Many parameter values can optionally refer to variables.

IBM Z Monitoring Configuration Manager supports the same variables as PARMGEN.

Tip: Instead of using variables, consider using LPAR-specific RTEDEF members.

Variables versus LPAR-specific RTEDEF members

Variables enable you to reuse a configuration profile member for different LPARs where LPARs require different parameter values.

However, using variables adds a precursor step to runtime environment started tasks. The step resolves variable values. The additional processing delays runtime environment startup.

LPAR-specific RTEDEF members, introduced by Monitoring Configuration Manager, offer an alternative to using variables for LPAR-specific parameter values.

Using LPAR-specific RTEDEF members instead of variables removes the variable-resolution precursor step from started tasks.

If you use LPAR-specific RTEDEF members instead of variables, started tasks are simpler and runtime environments start faster.

Using variables

To use variables, you must set the **RTE_SYSV_SYSVAR_FLAG** parameter to Y.

Variables, like parameters, are defined using name-value pairs and are stored in members of the RTEDEF library.

The following example, without variables, sets the parameter named **RTE_TCP_PORT_NUM** to the literal value 1918:

```
RTE_TCP_PORT_NUM 1918
```

The following example sets the parameter to the value of the variable **RTE_PORT**:

```
RTE_TCP_PORT_NUM &RTE_PORT.
```

Suppose your sysplex contains two LPARs: ZOS1 and ZOS2. In general, these LPARs have similar runtime environment configurations. However, on ZOS1 you want the monitoring server to listen on port 1918, whereas on ZOS2 you want the monitoring server to listen on port 1919.

In the variables configuration profile member for LPAR ZOS1, VAR$ZOS1, you set the **RTE_PORT** variable to 1918:

```
RTE_PORT 1918
```

In VAR$ZOS2, you set **RTE_PORT** to 1919:

```
RTE_PORT 1919
```

RTEDEF members that define variables

In the LPARs column of the following table, *Current* means: the LPAR on which the **GENERATE** action is performed.

Table 11. RTEDEF members that define variables, and the LPARs to which they apply		
Member name	**LPARs**	**Description**
VAR$GLOB	All	Variables configuration profile.
VAR$*lpar*	Current	LPAR-specific variables configuration profile.

If a variable is defined in both VAR$GLOB and VAR$*lpar*, then the value in VAR$*lpar* is used.

Unique variable names

While Configuration Manager supports RTEs with system variables (i.e. RTE_SYSV_SYSVAR_FLAG=Y), it does not support cases where the parameter's value is a variable with the same name as the parameter itself.

For example:

```
RTE_USS_RTEDIR &RTE_USS_RTEDIR
```

has the same name for the parameter (RTE_USS_RETDIR) and the variable (&RTE_USS_RTEDIR), which is not allowed.

In this case, you must change the name of the variable. For example, notice the addition of "MY_" in the variable name below:

```
RTE_USS_RTEDIR &MY_RTE_USS_RTEDIR
```

During a MIGRATE action, Configuration Manager automatically renames such variables by adding an '_R' suffix. If the variable name is 31 characters long, it adds only the '_' suffix. If the variable name is 32 characters long, Configuration Manager only adds a comment with a warning in the respective VAR$GLOB RTEDEF member, indicating you will have to take an action to rename the variable.

Setting up security exits in your runtime environment

Security exits are required for your runtime environment. You can use the **CREATE** and **MIGRATE** actions to set up your security exits library, and use the **GENERATE** action to create the necessary runtime members.

You must set up a library for your runtime environment that contains the OMEGAMON and IBM Tivoli Monitoring-related product security exits (such as KOBSUPDT OMEGAMON K*pp*SUPDI exits, Tivoli Monitoring Services: Engine security exits, and external security exits).

The following points provide an overview of the configuration that is required in Configuration Manager for security exits in your runtime environment:

- A dedicated library must be allocated and populated with the security exits. The default name for the security exits library is *rte_plib_hilev*.*rte_name*.SECEXITS. (You can override the name of this library using the **KFJ_SECURITY_EXITS_LIB** parameter in the **CREATE** or **MIGRATE** action.) If you use the **CREATE** or **MIGRATE** action to allocate the library, it will be populated with default security exit members. You can also import existing security exit members if you are migrating your runtime environment from PARMGEN.
- A reference to the security exits library is required in the **RTE_X_SECURITY_EXIT_LIB** parameter located in member *rte_plib_hilev*.RTEDEF(*rte_name*).

After the security exits library has been set up using Configuration Manager, you can modify the security exit members as needed for your environment. You can then use the **GENERATE** action to rebuild and relink them.

Setting up security exits using CREATE

You can use the **CREATE** action to allocate the security exits library using the default name and populate it with an initial set of configuration profile members. You can also use the **KFJ_SECURITY_EXITS_LIB** parameter to specify another name for the security exits library. If the specified data set does not exist, it will be allocated and populated with the default security exit members. If the specified data set does exist, it will be populated with the default security members, but no existing member will be overwritten. The **CREATE** action also populates the required reference to the library in the **RTE_X_SECURITY_EXIT_LIB** parameter.

For more information about running the **CREATE** action, see "CREATE" on page 34.

Setting up security exits using MIGRATE

If you are migrating your runtime environment from PARMGEN, you can use the **MIGRATE** action to import the PARMGEN security exits into the new runtime environment. Like the **CREATE** action, the **MIGRATE** action allocates the *rte_plib_hilev*.*rte_name*.SECEXITS library (or, optionally, the library specified in the **KFJ_SECURITY_EXITS_LIB** parameter). The **MIGRATE** action also copies the security exits used by the PARMGEN environment to the specified security exits library. Because the migration also imports runtime environment configuration settings from the PARMGEN environment, the **RTE_X_SECURITY_EXIT_LIB** parameter will contain the name of the security exits library used by the PARMGEN environment; you must review this setting and update it to use the proper *rte_plib_hilev*.*rte_name*.SECEXITS library (if necessary) before running the **GENERATE** action.

Note: The security exits library used in PARMGEN is identified in *rte_hilev*.*rtename*.RKANSAMU and is not changed as a result of the **MIGRATE** action. For more information about the differences between PARMGEN and Configuration Manager, see "Comparison with PARMGEN" on page 5.

For more information about running the **MIGRATE** action, see "MIGRATE" on page 55.

Rebuild and relink security exits using GENERATE

The **GENERATE** action automatically performs the required tasks of rebuilding and relinking the security exits. The **GENERATE** action also provides an optional setting, **OPTION SECEXITS**, that allows you to perform the security exits tasks separately from the normal **GENERATE** workflow, which can save valuable CPU cycles.

For more information about running the **GENERATE** action, see “GENERATE” on page 45.

Related tasks

CREATE

The **CREATE** action creates an initial runtime environment definition that you can customize to match your requirements.

Using override embed members

With Monitoring Configuration Manager, you can use override embed members to provide and maintain customization for your runtime environments.

A Monitoring Configuration Manager configuration creates a set of files that get embedded in a number of the most commonly updated runtime members in the user libraries. These override embed members can be used to specify user-defined parameters and values that might otherwise be overwritten by the **GENERATE** action when maintenance or upgrades are performed, or to override existing values.

The override embed members are stored in the *embeds data set*. The default name for this data set is *rte_plib_hilev*.*rte_name*.EMBEDS, or you can use a customized name. You can use one data set per RTE or you can share a common data set across multiple RTEs.

When using Monitoring Configuration Manager, you can enable the use of override embed members when creating an RTE or for an existing RTE.

The following parameters provide support for using override embed members:

KFJ_USE_EMBEDS
: This Monitoring Configuration Manager parameter controls whether override embed members are enabled for the RTE. When set to Y on the initial **CREATE** or **MIGRATE** action when creating an RTE, Monitoring Configuration Manager sets up the embeds data set, populates it with supported override embed parameters (if applicable), and defines it to the RTE using the **RTE_X_OVERRIDE_EMBEDS_LIB** parameter. Valid values are Y and N. The default is N.

KFJ_EMBEDS_LIB
: This Monitoring Configuration Manager parameter identifies the data set that contains the override embed values for the RTE. Use a valid MVS data set name for the value.

RTE_X_OVERRIDE_EMBEDS_LIB
: This parameter specifies the name of the source library for override embed members for the RTE and is located in RTEDEF(*rte_name*). This parameter and value is set up automatically when the initial **CREATE** or **MIGRATE** action runs to create an RTE and **KFJ_USE_EMBEDS** is set to Y. It needs to be added manually if you decide to add override embed support to an existing RTE.

Example

An example of how to specify the override embed parameters is shown below. By default, the override embed support is disabled. However, this example shows that it is enabled and provides the data set name.

```
//KCIVARS DD *
ACTION           CREATE
RTE_NAME         DEMO
RTE_PLIB_HILEV   TEST1.TST
...
KFJ_USE_EMBEDS   Y                        * Y|N valid values
KFJ_EMBEDS_LIB   TEST1.TST.DEMO.MYEMBEDS  * override default EMBEDs library
```

Tip: For more information about override embed members, see PARMGEN topics Override embed members and Customizing the override embed members. The override embed parameters and values are the same regardless if Monitoring Configuration Manager or PARMGEN is used to configure your RTE. Note that customization of the override embed members is also the same except, whereas in PARMGEN it is done from the WCONFIG, in Monitoring Configuration Manager it is done in the embeds data set specified in **RTE_X_OVERRIDE_EMBEDS_LIB**.

Related tasks

CREATE

The **CREATE** action creates an initial runtime environment definition that you can customize to match your requirements.

MIGRATE
The **MIGRATE** action imports configuration settings from a runtime environment that is configured with PARMGEN to one that is configured with Configuration Manager.

Enable override embed members when creating an RTE

Use this procedure to enable override embed members when creating a runtime environment (RTE).

Before you begin

Read "Using override embed members" on page 133.

This task applies to creating an RTE using the Monitoring Configuration Manager "CREATE" on page 34 or "MIGRATE" on page 55 action. For more information, see "Creating your first, minimal runtime environment" on page 13.

About this task

Override embed members can be used to specify user-defined parameters and values that might otherwise be overwritten by the **GENERATE** action when maintenance or upgrades are performed, or to override existing values. Using override embed members for your runtime environment requires a data set (the *embeds data set*) that contains the override embed parameters and values.

You can enable the use of override embed members when creating your RTE by including the **KFJ_USE_EMBEDS** parameter set to Y in the KCIVARS DD statement for the **CREATE** or **MIGRATE** actions. With the inclusion of this setting, Monitoring Configuration Manager creates the embeds data set, populates the override embed parameters and values, and defines the data set to the RTE using the **RTE_X_OVERRIDE_EMBEDS_LIB** parameter.

For the **CREATE** action, by default, the embeds data set is populated with the override embed parameters and values for the products that are installed in the respective CSI used to build the RTE. For the **MIGRATE** action, this library will contain the embed parameters and values from the source PARMGEN installation (as specified by parameter **KFJ_MIGRATE_WCONFIG**).

The default embeds data set name is *rte_plib_hilev*.*rte_name*.EMBEDS. Optionally, you can include the **KFJ_EMBEDS_LIB** parameter to use a custom data set name rather than the default name. Use of this parameter allows you to use the same embeds data set and settings for multiple RTEs.

Note: If you specify the name of an existing data set using parameter **KFJ_EMBEDS_LIB**, its contents will not be overwritten.

Procedure

1. As part of the process to create an RTE, submit a job that performs the **CREATE** action or **MIGRATE** action with **KFJ_USE_EMBEDS** set to Y and, optionally, **KFJ_EMBEDS_LIB** set to a custom data set name.

   Example JCL:

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            CREATE | MIGRATE
RTE_NAME          <rte_name>
RTE_PLIB_HILEV    <rte_plib_hilev>
...
KFJ_USE_EMBEDS    Y                        * Y|N valid values
KFJ_EMBEDS_LIB    <embeds_data_set_name>    * override default EMBEDs library
/*
```

Figure 48. Example JCL to enable override embed members for a new RTE

Where *<rte_name>* and *<rte_plib_hilev>* specify the RTE, and *<embeds_data_set_name>* is the optional custom embeds data set name.

2. Complete the creation of the RTE. For more information, see "Creating or updating a runtime environment" on page 21.
3. Update the override embed members as needed.

Results

Monitoring Configuration Manager sets up the embeds data set, populates it with supported override embed parameters (if applicable), and defines it to the RTE using the **RTE_X_OVERRIDE_EMBEDS_LIB** parameter.

Enable override embed members for an existing RTE

Use this procedure to enable override embed members for an existing runtime environment (RTE).

Before you begin

Read "Using override embed members" on page 133.

This task assumes that you have an existing runtime environment (RTE) that was created using Monitoring Configuration Manager. For more information, see "Creating or updating a runtime environment" on page 21.

About this task

Override embed members can be used to specify user-defined parameters and values that might otherwise be overwritten by the **GENERATE** action when maintenance or upgrades are performed, or to override existing values. Using override embed members for your runtime environment requires a data set (the *embeds data set*) that contains the override embed parameters and values.

You can enable the use of override embed members after you have created your RTE by running the **CREATE** action with the **KFJ_USE_EMBEDS** parameter set to Y. With the inclusion of this setting, Monitoring Configuration Manager creates the embeds data set and populates it, by default, with the override embed parameters and values for the products that are installed in the respective CSI.

Note: Running the **CREATE** action after the RTE has been created does not affect existing settings.

The default embeds data set name is *rte_plib_hilev.rte_name*.EMBEDS. Optionally, you can include the **KFJ_EMBEDS_LIB** parameter to use a custom data set name rather than the default name. Use of this parameter allows you to use the same embeds data set and settings for multiple RTEs.

Note: If you specify the name of an existing data set using parameter **KFJ_EMBEDS_LIB**, its contents will not be overwritten.

After the **CREATE** action is performed, you manually define the override embed data set to use for the RTE using the **RTE_X_OVERRIDE_EMBEDS_LIB** parameter.

Procedure

1. Submit a job that performs the **CREATE** action with **KFJ_USE_EMBEDS** set to Y and, optionally, **KFJ_EMBEDS_LIB** set to a custom data set name.

   Example JCL:

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            CREATE
RTE_NAME          <rte_name>
RTE_PLIB_HILEV    <rte_plib_hilev>
...
KFJ_USE_EMBEDS    Y                        * Y|N valid values
KFJ_EMBEDS_LIB    <embeds_data_set_name>    * override default EMBEDs library
/*
```

   Figure 49. Example JCL to enable override embed members for an existing RTE

   Where *<rte_name>* and *<rte_plib_hilev>* specify the RTE, and *<embeds_data_set_name>* is the optional custom embeds data set name.

   This job sets up the embeds data set and, if applicable, populates it with supported override embed parameters.
2. Open the RTEDEF(*rte_name*) member and add the **RTE_X_OVERRIDE_EMBEDS_LIB** parameter with the embeds data set name.

   Example:

```
RTE_X_OVERRIDE_EMBEDS_LIB        <embeds_data_set_name>
```

3. Update the override embed members as needed.

Remote deployment scenario

In a remote deployment scenario, you must create a runtime environment on a specific LPAR (the configuration LPAR), package the runtime environment data sets using the **PACKAGE** action, transfer the data sets to the remote target system (target LPAR), and deploy or restore the packaged runtime environment data sets on the target LPAR using the **DEPLOY** action.

Before you begin

Review the following information before you implement a remote deployment:

- Make sure the SMP/E target libraries for TKANMOD and TKANCUS on the target LPAR system and TKANMOD are APF-authorized. APF authorization is needed to run the necessary actions on the target LPAR system.
- If you are using the default RTE_TYPE (that is, SHARING with SMP), make sure a copy of the SMP/E target libraries is available on the target LPAR system, using the value you specified in **GBL_TARGET_HILEV** of member RTEDEF(GBL$PARM).

  If you cannot share your SMP/E target libraries on the target LPAR system, you can use the action **BLDREMDS** in the utility flow TKANSAM(KFJMAINT) to build the respective TKANSAM, TKANCUS, and TKANMOD data sets and transfer them to the target LPAR system.
- The z/OS operating system versions on your configuration LPAR and target LPAR should be ideally at the same level. If this is not the case, you will have to customize the z/OS specific libraries, such as SCEELKED in RTEDEF(GBL$PARM) or RTEDEF(GBL$*lpar*), to handle this situation.

  For example, parameter **GBL_DSN_CEE_SCEELKED** pointing to the default SCEE.SCEELKED system library could point to the z/OS 2.4 version of the library in **GBL$lpar1** and the z/OS 2.5 version in **GBL$lpar2**, respectively.

  When generating the runtime environment locally, the respective **GBL$lparn** member will be used.

About this task

The list in the following procedure describes a sample sequence of steps to be performed for a remote deployment.

You cannot customize some parameters when you are creating a runtime environment for remote deployment. For more information, see “Parameters that cannot be customized for remote deployment” on page 138.

Tip: If you need to assemble and link elements (for example, when applying maintenance), you can use the **GENERATE** action with **OPTION RELINK**. For more information, see “RELINK” on page 50.

Procedure

1. For the configuration LPAR, run the **CREATE** action to create an initial RTEDEF data set that will contain the configuration settings of your target LPAR.

   (Optional) Specify the **KFJ_LOCAL_PLIB_HILEV** value in the KCIVARS DD statement to indicate that you want to use different high-level qualifiers or z/OS UNIX System Services paths on the configuration LPAR and the target LPAR. The **KFJ_LOCAL_PLIB_HILEV** parameter is not needed if you will use the same HLQ on both the configuration LPAR and the target LPAR. See the “CREATE” on page 34 action for more details.
2. For the target LPAR, run the **DISCOVER** action to discover the subsystems and system symbols. This action will also create a RTEDEF data set that will contain the **Kpp@lpar** members for the subsystems discovered as well as the **SYS@lpar** member containing the system symbols.

3. For both the configuration LPAR and the target LPAR, transfer the RTEDEF created on the target LPAR to the configuration LPAR and merge the contents into the RTEDEF created in step 1 on the configuration LPAR.
4. For the configuration LPAR, customize your RTE as per your needs, understanding that the customizations will reflect the target LPAR system, such as the HLQs needed and features enabled in RTEDEF(Kpp$PARM or Kpp$lpar) members.

   (Optional) If you used the **KFJ_LOCAL_PLIB_HILEV** parameter, member RTEDEF(PCK$PARM) will be created, which allows you to map local Qshell and z/OS UNIX paths for allocating the runtime environment data sets or z/OS UNIX directories on the configuration LPAR.

   Note: The settings in RTEDEF(PCK$PARM) are applicable for all RTEs configured in the RTEDEF data sets, that is, for all RTEs of the respective SYSPLEX. If you want to use a different local HLQ for a specific target LPAR system on the configuration LPAR, you can create member RTEDEF(PCK$lpar) by copying RTEDEF(PCK$PARM) and making the respective changes.
5. For the configuration LPAR, run the **GENERATE** action for your target LPAR RTE by adding the **KFJ_SYSNAME** parameter to the KCIVARS DD statement. The value of **KFJ_SYSNAME** specifies the SYSNAME or LPAR name or the SYSSMFID if the LPAR name is longer than four characters.

   See "KFJ_SYSNAME" on page 99 for more details.
6. (Optional) For the configuration LPAR, if you used the **KFJ_LOCAL_PLIB_HILEV** parameter, after the **GENERATE** action completes, your runtime environment data sets are created using the HLQ or z/OS UNIX root path name mapping as specified in RTEDEF(PCK$PARM or PCK$lpar). The members RTEDEF(PCK$PARM) and RTEDEF(PCK$LPAR) will be read and used during RTE generation.

   Note: In this case, the runtime environment will not be able to start up as the target LPAR settings are used to generate the respective configuration settings.
7. For the configuration LPAR, run the **PACKAGE** action to build transferable dump data sets. Refer to "PACKAGE" on page 59 for more details about the data sets created and the options that can be used.
8. For the configuration LPAR and the target LPAR, transfer the dump data sets to the target LPAR using the procedure of your choice (for example, FTPS).
9. For the target LPAR, run the **DEPLOY** action to unpack or restore the (tersed) data sets. See "DEPLOY" on page 61 for more details and the options that can be used.
10. For the target LPAR, adjust or copy your STC procedures.

Related tasks

PACKAGE
The **PACKAGE** action packages a runtime environment that can then be deployed to a remote system.

DEPLOY
The **DEPLOY** action deploys a packaged runtime environment to a remote system.

Parameters that cannot be customized for remote deployment

You cannot customize some parameters when you are creating a runtime environment for remote deployment.

The following table lists the parameters that you cannot customize when you are creating a runtime environment for remote deployment:

Table 12. Parameters that cannot be customized for a remote deployment runtime environment

Product or component	Parameter
Global parameters	GBL_USER_JCL
IBM Tivoli Composite Application Manager (ITCAM) for Application Diagnostics Agent	KYN_XAI01_SUBAGENT_PRODHOME

Table 12. Parameters that cannot be customized for a remote deployment runtime environment (continued)

Product or component	**Parameter**
IBM OMEGAMON for Messaging on z/OS	KQI_HFS_HFSROOT_DIR1
IBM OMEGAMON for Storage on z/OS	KS3_APP_ZFS_DIR
IBM OMEGAMON for Db2 Performance Expert[1]	KD2_OMPE_DSHLQ KD2_OMPE_UNIT KD2_OMPE_VOLUME KD2_OMPE_STOCLAS KD2_OMPE_MGMTCLAS KD2_OMPE_VSAM_DSHLQ KD2_OMPE_VSAM_VOLUME KD2_OMPE_VSAM_STOCLAS KD2_OMPE_VSAM_MGMTCLAS KD2_PF*nn*_HIS_DYN_DSNAME KD2_PF*nn*_HIS_DYN_VOLUME KD2_PF*nn*_HIS_DYN_UNIT KD2_PF*nn*_HIS_DYN_SCLAS KD2_PF*nn*_HIS_DYN_MCLAS KD2_PF*nn*_HIS_SEQ_ARC_DS KD2_PF*nn*_HIS_SEQ_ARC_VOLU KD2_PF*nn*_HIS_SEQ_ARC_UNIT KD2_PF*nn*_HIS_SEQ_ARC_SCLA KD2_PF*nn*_HIS_SEQ_ARC_MCLA KD2_PF*nn*_HIS_GDG_DSNAME KD2_PF*nn*_HIS_GDG_VOLUME KD2_PF*nn*_HIS_GDG_UNIT KD2_PF*nn*_HIS_GDG_SCLAS KD2_PF*nn*_HIS_GDG_MCLAS KD2_PF*nn*_AEXCP_D2TPTDSN KD2_PF*nn*_AEXCP_D2TPLDSN KD2_PF*nn*_AEXCP_D2TPFDSN KD2_PF*nn*_AEXCP_D2TPVL KD2_PF*nn*_AEXCP_D2TPTFSC KD2_PF*nn*_AEXCP_D2TPTFMC
IBM OMEGAMON for Db2 Performance Expert 5.5.0[1]	KD2_PF*nn*_HIS_VSAM_VOLUME KD2_PF*nn*_HIS_VSAM_SCLAS KD2_PF*nn*_HIS_VSAM_MCLAS KD2_PF*nn*_HIS_SEQ_VOLUME KD2_PF*nn*_HIS_SEQ_UNIT KD2_PF*nn*_HIS_SEQ_SCLAS

Table 12. Parameters that cannot be customized for a remote deployment runtime environment (continued)	
Product or component	**Parameter**
IBM OMEGAMON for Db2 Performance Expert 5.4.0[1]	KD2_PF*nn*_HIS_LOG*n* KD2_PF*nn*_HIS_VSAM_VOLUME*n* KD2_PF*nn*_HIS_VSAM_SCLAS*n* KD2_PF*nn*_HIS_VSAM_MCLAS*n* KD2_PF*nn*_HIS_SEQLOG*x* KD2_PF*nn*_HIS_SEQ_VOLUME*x* KD2_PF*nn*_HIS_SEQ_UNIT*x* KD2_PF*nn*_HIS_SEQ_SCLAS*x* KD2_PF*nn*_HIS_SEQ_MCLAS*x* where *x* is 1 to 7

[1] The list of parameters for IBM OMEGAMON for Db2 Performance Expert cannot be customized if **KFJ_LOCAL_KD5_RUN_ALLOC** is specified with the value GENERATE. If you want to customize these parameters, then use the value DEPLOY or NONE.

Using SMP/E target library copies

Use the Configuration Manager *target copy* feature to create one or more copies of your SMP/E target libraries, from which you can create or update your runtime environments.

Tip: If you are moving from PARMGEN to Configuration Manager, the Configuration Manager target copy feature provides an alternative to the PARMGEN base library feature.

Define SMP/E target copy settings

Create a member in your RTEDEF library to contain your SMP/E target copy settings, and update the settings as needed.

Before you begin

The Configuration Manager target copy feature is used to create and maintain one or more copies of your SMP/E target libraries, from which you can create or update your runtime environments.

The first step in setting up the use of an SMP/E target library copy is to create a dedicated member in your RTEDEF library that will contain the target copy settings. To review the initial content of the SMP/E target copy member that will be created in this task, see “RTEDEF(trg_copy_name)” on page 116.

About this task

You use the **CREATE** action with option **TRGCOPY** to create and initially populate a member in an RTEDEF library specifically for SMP/E target copy settings.

The **CREATE** action supports the following settings when creating the SMP/E target copy member:

OPTION TRGCOPY
Creates a target copy member inside the RTEDEF library. The member is named using the **TRG_COPY_NAME** parameter.

TRG_COPY_NAME
Specifies the name of the member to create in the RTEDEF data set to contain the SMP/E target copy settings. If the specified member already exists, it is not overwritten.

TRG_COPY_HILEV
(Optional) Specifies the high-level qualifier for the target copy data sets. If not specified, the **RTE_PLIB_HILEV** parameter value is used.

RTE_PLIB_HILEV
Specifies the high-level qualifier of the RTEDEF library; if the RTEDEF data set does not exist, it is created. This parameter value is also used as the default value for the high-level qualifier for the target copy data sets if **TRG_COPY_HILEV** is not specified.

After the member has been created, you must then review and update the settings, as needed. For more information about the created member, see “RTEDEF(trg_copy_name)” on page 116.

Procedure

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a **CREATE** action.
2. Specify **OPTION TRGCOPY**.
3. Specify a value for required parameter **TRG_COPY_NAME** to use as the name of the target copy member.
4. Specify the high-level qualifier of the RTEDEF in parameter **RTE_PLIB_HILEV**.
5. (Optional) Specify a value for parameter **TRG_COPY_HILEV** to use as the high-level qualifier for the target copy data sets.
6. Run the KFJJMCM job to create the SMP/E target copy member in the RTEDEF data set.

Job messages for the **CREATE** action are written to the KCIPRINT SYSOUT data set.

7. Review and update the RTEDEF(*trg_copy_name*) member, as needed.

Example

The following JCL job creates the SMP/E target copy member MYCOPY in data set TSOUID.PROD.RTEDEF. If the specified RTEDEF data set does not exist, it is created. If the specified member already exists, it is not overwritten.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
*
ACTION                  CREATE
OPTION                  TRGCOPY
TRG_COPY_NAME           MYCOPY
RTE_PLIB_HILEV          TSOUID.PROD
/*
```

Figure 50. Example JCL to create the SMP/E target copy member

```
* High-level qualifier of SMP/E target libraries
GBL_TARGET_HILEV                MONSUITE

* SMP/E target directory containing TKANJAR files (KGW, KJJ)
GBL_USS_TKANJAR_PATH            "/usr/lpp/kan/bin/IBM"

TRG_COPY_NAME                   MYCOPY
* High-level qualifier of the copy of SMP/E target libraries
TRG_COPY_HILEV                  TSOUID.PROD
* Directory for a copy of SMP/E TKANJAR files (KGW, KJJ)
TRG_COPY_TKANJAR_PATH           "/var/rtehome/MYCOPY/kan/bin/IBM"

CONFIGURE_TEMS_KDS              Y  * TEMS
CONFIGURE_E3270UI_KOB           Y  * Enhanced 3270
CONFIGURE_CICS_KC5              Y  * CICS TS
CONFIGURE_CICS_TG_KGW           Y  * CICS TG
CONFIGURE_DB2_AGENT_KD5         Y  * Db2
CONFIGURE_IMS_KI5               Y  * IMS
CONFIGURE_JVM_KJJ               Y  * JVM
CONFIGURE_ZOS_KM5               Y  * z/OS
CONFIGURE_MESSAGING_KMQ         Y  * MQ
CONFIGURE_MESSAGING_KQI         Y  * Integration Bus
CONFIGURE_NETVIEW_KNA           Y  * Netview
CONFIGURE_MFN_KN3               Y  * Network
CONFIGURE_STORAGE_KS3           Y  * Storage
CONFIGURE_OMEGAVIEW_KWO         Y  * Integration Monitor
CONFIGURE_ITCAMAD_KYN           Y  * ITCAM for Applications
CONFIGURE_ACM_KRN               Y  * Advanced Catalog Mgmt
CONFIGURE_ARD_KRH               Y  * Advanced Rpt and Mgmt
CONFIGURE_AAD_KRG               Y  * Advanced Audit
CONFIGURE_AAM_KRJ               Y  * Advanced Alloc Mgmt
CONFIGURE_ATAM_KRK              Y  * Automated Tape Alloc
CONFIGURE_ABR_KRV               Y  * Advanced Backup and Rec
```

Figure 51. Example SMP/E target copy member

What to do next

Use the new target copy member to make a copy of your SMP/E target libraries. See “Copy SMP/E target libraries” on page 143.

Note: Make sure to carefully review your target copy settings before continuing with the next step.

Copy SMP/E target libraries

Make a copy of your SMP/E target libraries using your defined SMP/E target copy settings.

Before you begin

The Configuration Manager target copy feature is used to create and maintain a copy of your SMP/E libraries, from which you can create or update your runtime environments.

Before you can copy the libraries, you must have created and updated your SMP/E target copy settings, as described in "Define SMP/E target copy settings" on page 141. You can then use your defined settings to copy the libraries, as described in this task.

Important: Make sure you have carefully reviewed your target copy settings before continuing with this task.

About this task

You use the **GENERATE** action with option **TRGCOPY** to copy your SMP/E target libraries from the original source to a new set of data sets, as defined in your SMP/E target copy settings in member RTEDEF(trg_copy_name).

Data sets for the copy of the SMP/E target libraries are allocated with the name *trg_copy_hilev.trg_copy_name*. Target copy libraries are always non-VSAM. Only the files for the products installed in your environment, as specified by the **CONFIGURE_*** flags in the RTEDEF(*trg_copy_name*) member, are copied.

When used to copy SMP/E target libraries, the **GENERATE** action supports the following settings

OPTION TRGCOPY
Creates a copy of your SMP/E target libraries from the original source to a new set of data sets, as defined in member RTEDEF(trg_copy_name).

TRG_COPY_NAME
Specifies the name of the member in the RTEDEF data set that contains the SMP/E target copy settings.

RTE_PLIB_HILEV
Specifies the high-level qualifier of the RTEDEF library.

Procedure

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a **GENERATE** action.
2. Specify **OPTION TRGCOPY**.
3. Specify the name of the target copy member for required parameter **TRG_COPY_NAME**.
4. Specify the high-level qualifier of the RTEDEF in parameter **RTE_PLIB_HILEV**.
5. Run the KFJJMCM job to copy your SMP/E target libraries from the original source to a new set of data sets.

   Job messages for the **GENERATE** action are written to the KCIPRINT SYSOUT data set.

Example

The following JCL job copies SMP/E target libraries from the original source to a new set of data sets, as defined in the SMP/E target copy member MYCOPY in data set TSOUID.PROD.RTEDEF. In this example, new data sets are allocated using the high-level qualifier TSOUID.PROD.MYCOPY. The number of data sets copied depends on the products selected for configuration in the target copy member.

Important: The high-level qualifier for the data set name specified in the KCIFLOW DD statement must match the **GBL_TARGET_HILEV** parameter value that is specified in the RTEDEF(*trg_copy_name*) member.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            GENERATE
OPTION            TRGCOPY
TRG_COPY_NAME     MYCOPY
RTE_PLIB_HILEV    TSOUID.PROD
/*
```

Figure 52. Example JCL to copy SMP/E target libraries

What to do next

You can use the copy of the SMP/E target libraries to create or update your runtime environments.

Create a target copy for an existing runtime environment

Use this procedure to configure an existing runtime environment to use a copy of SMP/E target libraries.

Before you begin

This procedure builds upon information provided in the following topics:

- "Define SMP/E target copy settings" on page 141
- "Copy SMP/E target libraries" on page 143

You can review these topics for additional details.

About this task

The following procedure provides the steps to update your existing runtime environment to use a copy of SMP/E target libraries instead of original SMP/E target libraries.

Procedure

1. Create the target copy member by running the **CREATE** action with **OPTION TRGCOPY** and the following additional settings:
   - Include the **RTE_PLIB_HILEV** parameter, which must point to an existing RTEDEF data set.
   - Include the **TRG_COPY_NAME** parameter, which must specify a new member name, one that does not exist in the RTEDEF data set.

   For more details, see "Define SMP/E target copy settings" on page 141.
2. After new member RTEDEF(*trg_copy_name*) has been created, review the settings in the member and update as needed.

   Note: Make sure to review parameter **TRG_COPY_HILEV** in the new member, which specifies the high-level qualifier for the SMP/E target library copy.
3. Create the copy of the SMP/E target libraries by running the **GENERATE** action with **OPTION TRGCOPY** and the following additional settings:
   - Include the **RTE_PLIB_HILEV** parameter, which must point to an existing RTEDEF data set.
   - Include the **TRG_COPY_NAME** parameter, which must specify the name of the newly created member.

   For more details, see "Copy SMP/E target libraries" on page 143.
4. After the **GENERATE** action has completed and the SMP/E target libraries have been copied to the new location, make the following updates:

a) Update parameter **GBL_TARGET_HILEV** in member RTEDEF(GBL$PARM) or RTEDEF(GBL$*lpar*) to point to the newly created copy of the SMP/E target libraries.

b) If parameter **GBL_USS_TKANJAR_PATH** parameter is required for your runtime environment, update the **GBL_USS_TKANJAR_PATH** parameter to use the value from **TRG_COPY_TKANJAR_PATH** in member RTEDEF(*trg_copy_name*).

5. Run the **GENERATE** action for the modified runtime environment.
6. Perform standard post-configuration steps, such as copying generated started tasks into a site-specific location.
7. (Optional) You can repeat steps 1 through 3 to create multiple copies of your SMP/E target libraries, which you can use to apply a staged rollout of maintenance for an existing runtime environment.

Maintain SMP/E target library copies

Keep the copy of your SMP/E target libraries up to date.

Before you begin

The Configuration Manager target copy feature is used to create and maintain one or more copies of your SMP/E target libraries, from which you can create or update your runtime environments.

After you have created a copy of your SMP/E target libraries using the target copy feature, as described in "Copy SMP/E target libraries" on page 143, you can refresh the copy as needed, as described in this task.

About this task

You use the **GENERATE** action with option **TRGCOPY** to refresh the copy of your SMP/E target libraries.

Procedure

To keep the SMP/E target copy in sync with your original SMP/E target libraries, simply run the **GENERATE** action with option **TRGCOPY** again. It will refresh all the libraries and will copy all the required members from your SMP/E target libraries to a copy of the SMP/E target libraries.

Troubleshooting

Use these topics to troubleshoot issues with Monitoring Configuration Manager.

How to navigate Configuration Manager action output

Use the method described in this procedure to navigate Configuration Manager output when troubleshooting a problem.

Before you begin

Configuration Manager writes job output for the **KCIOMEGA** actions to a number of output data sets. Output from invoked utilities is also generated, with each utility writing to its own output data set.

Note: KCIALPHA is an APF-authorized version of **KCIOMEGA**. The information in this topic also applies when using **KCIALPHA**.

The following table lists the output data sets that Configuration Manager generates and retains.

Table 13. Standard sysout data sets

Output data set	Description
KCIPRINT	**KCIOMEGA** program messages and messages about Configuration Manager processing
KCITRACE	**KCIOMEGA** workflow trace output, used primarily by IBM Software Support for troubleshooting
KCIVARSO	KCIVARS input that was used in the JCL
DSNPROUT	Contents of the RTEDEF data set
MIGRPT	Output for the **MIGRATE** action
$REPORT	Output for the **DEPLOY** action
JESMSGLG, JESJCL and JESYSMSG	Standard JES-produced output

In addition to the output data sets listed in Table 13 on page 147, Configuration Manager also generates other output that it does not retain and deletes from the spool automatically.

If a Configuration Manager job fails, the DD statement that contains the error message is retained and printed.

Notes:

- JES3 does not allow deleting DD output. If your site uses JES3, you might see additional output, which can be considerable.
- With APAR OA65222, the **DEBUG** action option (**OPTION DEBUG**), is introduced. This option generates additional output that is otherwise suppressed by default. This option should only be used under the guidance of IBM Software Support. Prior to this enhancement, all generated output was retained and printed to the spool, which could result in hundreds of output data sets.

About this task

When troubleshooting an issue with Configuration Manager jobs, use the steps in the following procedure to navigate the Configuration Manager output.

To view Configuration Manager output, it is recommended that you use SDSF and the **?** action character to list the output data sets. Consider sorting by DSID in SDSF to get the proper order.

Since most problems cause the Configuration Manager action to immediately stop, it is likely that the last utility invoked is the cause of the problem.

Note: For most problems, you do not need to examine the KCITRACE sysout data set as it is typically only required when there is a logic error in the workflow. KCITRACE is mostly used by IBM Software Support.

Procedure

1. View the KCIPRINT sysout data set, and look for the error message that explains why the problem occurred.
2. If the problem was caused by a utility, then scroll down to the end of the data set list and select the last file.

   This is typically the output from the failed utility and should provide more information about the problem.
3. In the event of a system problem (for example, an abend), then view JESMSGLG (the JES message log) and look for abnormal system messages.

Configuration Manager output data sets

Review the types of output data sets that are produced by Configuration Manager.

To troubleshoot issues with Configuration Manager jobs, use a tool such as SDSF to view the JES output data sets.

KCIPRINT sysout data set

Look at KCIPRINT first. KCIPRINT contains messages from the KCIOMEGA program interspersed with messages about Configuration Manager processing.

Here is an example KCIPRINT for a successful job:

```
KFU00001I KCIOMEGA is starting; SYSPLEX=sysplex LPAR=lpar DATE=…
KFU00002I INVOKE processing is about to commence; MEMBER=KFJOMEGA
…
Workflow has completed successfully
KFU00004I KCIOMEGA is ending; RC=rc SYSPLEX=sysplex LPAR=lpar DATE=…
```

Figure 53. Example KCIPRINT output data set for a successful Configuration Manager job

KCITRACE sysout data set

KCITRACE contains the KCIPRINT contents, and, in addition, also includes the source of each workflow skeleton and additional messages.

KCITRACE records that start with two consecutive plus signs (++) show the previous record after variable substitution. For example:

```
//RTEDEF DD DSN=%RTEDEF_DSN%,PASS=YES
++RTEDEF DD DSN=MYID.MONITORS.RTEDEF,PASS=YES
```

Renamed SYSPRINT sysout data sets

The KCIOMEGA program runs a workflow, such as the **GENERATE** action of Configuration Manager, that can invoke many programs, resulting in long job output listings. All of these programs run in the same job step as KCIOMEGA.

Many programs write to the ddname SYSPRINT. To avoid a job output listing with multiple SYSPRINT ddnames for the same step name, KCIOMEGA renames SYSPRINT sysout data sets to match the

corresponding step name in the workflow skeleton shown in KCITRACE. This makes it easier to find the SYSPRINT for each step in the skeleton.

Parameter values used

The KCIPRINT sysout data set from a **GENERATE** action contains an ordered list of the RTEDEF library members that the action uses. For example:

```
01. Using parameters in rte_plib_hilev.RTEDEF(rte_name)
02. Using parameters in rte_plib_hilev.RTEDEF(KDS$PARM)
03. Using parameters in rte_plib_hilev.RTEDEF(GBL$PARM)
```

The corresponding KCITRACE sysout data set contains an alphabetical list of parameters and symbols (such as workflow variables) with their values. The list is preceded by the following heading:

```
>Parameter and symbol values
```

Note to users with PARMGEN experience: The **GENERATE** action of Configuration Manager creates an *rte_plib_hilev.rte_name*.WCONFIG(*rte_name*) member that is similar to the member created by PARMGEN, with one key difference: the member created by Configuration Manager contains *default* parameter values. It does *not* reflect the values in your RTEDEF library members. To see the parameter values used by Configuration Manager, refer to the `>Parameter and symbol values` heading in the KCITRACE sysout data set. As such, the WCONFIG data sets as a whole should be considered as a *black box*.

Parameter validation report ($VALRPT) sysout data set

If the set of parameters that you specified in the RTEDEF library is invalid (for example, a required parameter is missing or a value is incorrect), then the sysout data sets include $VALRPT. This data set contains the same parameter validation report generated by PARMGEN, with the difference that you do not have to manually navigate to the data set and open it.

Review the report, correct the parameters, and then resubmit the job.

Problem determination data collection (PDCOLLECT)

Use the PDCOLLECT utility to collect diagnostic information.

Before you begin

To use the Problem Determination Data Collection (PDCOLLECT) utility, you must install SDSF and be licensed to use it. If you do not have SDSF, you can copy the complete address space logs, including the JES output, RKLVLOG, RKPDOUT, RKPDLOG, SYSPRINT, and so on, into a data set. The DCB information for the output data set can be:

```
Organization: PS
Record format: VB
Record length: 240
Block size: 27998
```

About this task

The Configuration Manager PDCOLLECT utility action collects data that includes the following:

- System configuration
- Network information
- Self-describing agent (SDA) information
- Configuration Manager-related information (RTEDEF and EMBEDS data sets)
- Run Time Environment (RTE) data sets

- Output from the specified job

Note: This feature is available with APAR OA62230 (PTF UJ06864).

Procedure

1. Locate the sample JCL job in the SMP/E target library *tHilev*.TKANSAM(KFJMAINT).
2. Specify the following parameters in the KCIVARS DD statement:
   a) Use **ACTION PDCOLLECT**.
   b) Specify the **RTE_PLIB_HILEV** and **RTE_NAME** parameters that will point to the required RTEDEF data set.
   c) If the OMEGAMON address space logs are in the SDSF output queue, specify the **KFJ_PDCOL_JOB_NAME** and **KFJ_PDCOL_JOB_ID** parameters.
   d) If your OMEGAMON address space logs were copied to a sequential data set, replace the **KFJ_PDCOL_JOB_NAME** and **KFJ_PDCOL_JOB_ID** parameters with the **KFJ_PDCOL_JOB_OUTPUT** parameter, and specify the data set name.
3. Submit the updated JCL.

Results

The file &SYSUID.KCIPDCOL.PDCOLPDS.TRS is generated. You can FTP the output data set to the IBM Support Center.

If you have to redirect your results to a different high-level qualifier, specify the **KFJ_PDCOL_HLQ** parameter in KCIVARS DD statement.

Example

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//S1 EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB DD DISP=SHR,DSN=<tlib_hlq>.TKANMOD
//KCIFLOW DD DISP=SHR,DSN=<tlib_hlq>.TKANCUS(KFJMAINT)
//KCIVARS DD *
ACTION                 PDCOLLECT
RTE_NAME               RTEZOS1
RTE_PLIB_HILEV         TSOUID.MONSUITE

* For PDCOLLECT: following parameters are required

KFJ_PDCOL_JOB_NAME     OMEGDS
KFJ_PDCOL_JOB_ID       S654321
```

Messages

Use the information in these messages to help you diagnose and solve problems running Monitoring Configuration Manager jobs.

Message format

Monitoring Configuration Manager message identifiers have the following format:

KF*xnnnnns*

where:

KF*x*
Origin of the message:

KFJ
A Monitoring Configuration Manager workflow.

KFU
The underlying KCIOMEGA program or its APF-authorized version, KCIALPHA. KCIOMEGA is the job template engine that runs Monitoring Configuration Manager workflows.

nnnnn
5-digit message identification number.

s
Severity of the message:

I
Informational.

W
Warning to alert you to a possible error condition.

E
Error. Workflow processing typically stops.

The documentation for each message includes the following information:

Explanation
Describes what the message text means, why the message occurred, and what its variables represent.

System action
Describes what the system will do in response to the event that triggered this message.

User response
Describes whether a response is necessary, what the appropriate response is, and how the response will affect the system or program.

KFJ messages

Messages with the prefix KFJ are from Monitoring Configuration Manager workflows.

Many messages from Monitoring Configuration Manager workflows are self-explanatory and do not begin with an identifier. Only the messages that require further explanation have an identifier.

KFJ00001E K*pp* version *version* is not supported by this configuration tool

Explanation:
IBM Z Monitoring Configuration Manager checks the installed versions of products to be configured in the runtime environment.

System action:
No action is performed. The job ends.

User response
In the JCL that runs Monitoring Configuration Manager, specify a KCIFLOW DD statement that refers to an

installation containing product versions supported by Monitoring Configuration Manager.

If you have only earlier product versions that are not supported by Monitoring Configuration Manager, then consider upgrading.

KFJ00002I **K*pp* version *version* is not supported by this configuration tool**

Explanation:
IBM Z Monitoring Configuration Manager checks the installed versions of products in an SMP/E installation target library. The K*pp* product version cannot be configured using Monitoring Configuration Manager.

System action:
The system proceeds with job execution.

User response

In the JCL that runs Monitoring Configuration Manager, specify a KCIFLOW DD statement that refers to an installation containing product versions supported by Monitoring Configuration Manager.

If you have only earlier product versions that are not supported by Monitoring Configuration Manager, then consider upgrading.

KFJ00003E **CONFIGURE_*agent* is set to Y but K_*pp*_ is not installed.**

Explanation:
The **GENERATE** action found the CONFIGURE_*agent* parameter set to "Y" in RTEDEF, but this version of the K_*pp* agent is not installed.

System action:
The **GENERATE** action stops before generating runtime members.

User response:
In the JCL that runs Monitoring Configuration Manager, verify that the correct installation data set is specified. Set CONFIGURE_*agent* to "N" in RTEDEF.

KFJ00004E **CONFIGURE_*agent* is set to Y but K_*pp*_ installed version is unknown.**

Explanation:
The **GENERATE** action found the CONFIGURE_*agent* parameter set to "Y" in RTEDEF, but could not determine the K_*pp* version of the agent.

System action:
The **GENERATE** action stops before generating runtime members.

User response:
In the JCL that runs Monitoring Configuration Manager, verify that the correct installation data set is specified. Set CONFIGURE_*agent* to "N" in RTEDEF.

KFJ00005E **RTE NAMES do not match.**

Explanation:
The MIGRATE action has not found the specified RTE in the source WCONFIG data set.

System action:
The MIGRATE action stops.

User response:
Verify that the RTE_NAME and KFJ_MIGRATE_WCONFIG values are correct.

KFJ00006E ***rte_name* has not been found in *kfj_migrate_wconfig***

Explanation:
The specified RTE member, *rte_name*, has not been found in the migrate source WCONFIG data set listed in the message.

System action:
The MIGRATE action stops.

User response:
Verify that the RTE_NAME and KFJ_MIGRATE_WCONFIG values are correct.

KFJ00007E **RTEDEF dataset *rtedef* already exists.**

Explanation:
Specified target RTEDEF data set, listed as *rtedef* in the message, already exists so it cannot be created again.

System action:
The MIGRATE action stops.

User response:
Use a different RTE_PLIB_HILEV or specify CONFIRM=Y in the job to overwrite the existing RTEDEF data set.

KFJ00008W **Agent *Kpp* is not installed in the target SMP/E**

Explanation:
The MIGRATE action has detected that the agent *Kpp* is set to CONFIGURE_*agent*_*Kpp*=Y in the source WCONFIG, but it is not installed in the target SMP/E environment.

System action:
The MIGRATE action sets CONFIGURE_*agent*_*Kpp*=N and continues.

User response:
Verify that the correct SMP/E environment has been set in the job running the MIGRATE action.

KFJ00200E **Parameter required but not specified: *parameter***

Explanation:
Before generating runtime members, the **GENERATE** action checks whether this required parameter has been specified.

System action:
The **GENERATE** action stops before generating runtime members.

User response:
Specify the required parameter in the appropriate RTEDEF member, and then resubmit the job.

KFJ00201E **GBL_USS_TKANJAR_PATH directory *path* does not exist**

Explanation
The **GBL_USS_TKANJAR_PATH** parameter specifies the path of the target z/OS UNIX System Services directory that is defined in the SMP/E installation jobs by ddname TKANJAR.

Before generating runtime members, the **GENERATE** action checks whether this directory exists.

System action:
The **GENERATE** action stops before generating runtime members.

User response
Specify the correct path in the appropriate RTEDEF member, and then resubmit the job.

If you do not know the path, contact the person who installed the products.

The default path for the TKANJAR ddname is `/usr/lpp/kan/bin/IBM`.

KFJ00202E **RTE_USS_RTEDIR must not be a subdirectory of GBL_USS_TKANJAR_PATH**

Explanation
The **RTE_USS_RTEDIR** and **GBL_USS_TKANJAR_PATH** parameters each specify the path of a z/OS UNIX System Services directory. **GBL_USS_TKANJAR_PATH** is an SMP/E target directory, or a copy. **RTE_USS_RTEDIR** specifies where to generate runtime members; some runtime environment started tasks also write to files under this directory.

RTE_USS_RTEDIR must not be a subdirectory of **GBL_USS_TKANJAR_PATH** because SMP/E target directories and their descendants should be read-only for most users. However, some OMEGAMON products write to files under the **RTE_USS_RTEDIR** directory.

Before generating runtime members, the **GENERATE** action checks whether **RTE_USS_RTEDIR** is a subdirectory of **GBL_USS_TKANJAR_PATH**.

System action:
The **GENERATE** action stops before generating runtime members.

User response:
Change the value of **RTE_USS_RTEDIR**, and then resubmit the job.

KFJ00203E **User requires write access to the *zos_unix_path_parameter* directory *path***

Explanation
The user who runs the job that performs the **GENERATE** action must be able to write to the z/OS UNIX System Services directory specified in the parameter, where *zos_unix_path_parameter* is **RTE_USS_RTEDIR** or **TRG_COPY_TKANJAR_PATH**, and *path* is the directory path.

The **RTE_USS_RTEDIR** parameter specifies the path of the z/OS UNIX directory where the **GENERATE** action writes runtime members.

The **TRG_COPY_TKANJAR_PATH** parameter specifies the path of the z/OS UNIX directory where the **GENERATE** action with option **TRGCOPY** copies SMP/E installation files.

System action:
The **GENERATE** action stops before generating runtime members.

User response
Follow your local site practices to grant the user write access to the directory.

For example, set the directory permissions to 775. The following z/OS UNIX shell command sequence (requires superuser for **chmod**) recursively sets the permissions of `/var/rtehome` and its descendants:

```
echo chmod -R 775 /var/rtehome | su
```

KFJ00204E **An error occurred creating RTE_USS_RTEDIR directory *path***

Explanation:
The **RTE_USS_RTEDIR** parameter specifies the path of the z/OS UNIX System Services directory where the **GENERATE** action writes runtime members. If this directory does not exist, then the user who runs the job that performs the **GENERATE** action must be able to create this z/OS UNIX directory.

System action:
The **GENERATE** action stops before generating runtime members.

User response:

Follow your local site practices to grant the user write access to create the directory.

KFJ00205E [GBL_USS_TKANJAR_PATH | KFJ_LOCAL_USS_TKANJAR_PATH] directory *path* is empty

Explanation:
The z/OS UNIX System Services directory specified in the **GBL_USS_TKANJAR_PATH** or **KFJ_LOCAL_USS_TKANJAR_PATH** parameter must contain installation files.

System action:
The **GENERATE** action stops before generating runtime members.

User response:
Verify if the **GBL_USS_TKANJAR_PATH** or **KFJ_LOCAL_USS_TKANJAR_PATH** parameter is specified correctly.

KFJ00205W The length of SYSNAME exceeds 4.

Explanation:
Before generating members in the RTEDEF library, the DISCOVERY, GENERATE, and MIGRATE actions verify if the SYSNAME parameter exceeds 4 characters in length.

System action

The system action depends on whether KFJ_SYSNAME is specified.

- If KFJ_SYSNAME is not specified in the KCIVARS DD statement in the JCL, the system will use the SYSSMFID parameter instead of SYSNAME.
- If KFJ_SYSNAME is specified and it does not exceed 4 characters in length, the system will use the KFJ_SYSNAME value instead of SYSNAME.

User response

If you want to change the SYSNAME system parameter, specify the KFJ_SYSNAME parameter in the KCIVARS DD statement in the JCL **before** submitting DISCOVERY and GENERATE actions. See the example below.

```
//UID#ZMCM
JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1 EXEC PGM=KCIALPHA,REGION=0M,DYNAMNBR=256
//STEPLIB DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW DD
DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS DD *
ACTION          DISCOVER
RTE_NAME        RTE1
RTE_PLIB_HILEV  TSOUID.MONSUITE
KFJ_SYSNAME     MVS1
/*
```

KFJ00206E RTE_X_SECURITY_EXIT_LIB data set *name* is not allocated

Explanation:
The data set specified in parameter **RTE_X_SECURITY_EXIT_LIB** is not allocated. Typically, this data set is allocated during the **CREATE** or **MIGRATE** action.

System action:
The **GENERATE** action stops before generating runtime members.

User response:
Verify that the **RTE_X_SECURITY_EXIT_LIB** parameter is correctly specified in the RTEDEF(*rte_name*) member. Ensure that the **RTE_X_SECURITY_EXIT_LIB** data set is allocated and contains the required security exits.

KFJ00207E Full discovery not attempted - APF authorization required

Explanation:
The KCIALPHA load library has been used for the DISCOVER action, but it is not APF authorized.

System action:
KCIALPHA performed a partial discovery.

User response:
Partial discovery was still performed and relevant members might have been created in RTEDEF. These members will not be overwritten by a full discovery process. If full discovery is required, APF authorize the KCIALPHA load library and rerun the job.

KFJ00208E User requires read access to the *name* directory

Explanation:
The user who runs the job that performs the **PACKAGE** action must be able to read this z/OS UNIX System Services directory.

System action:
The **PACKAGE** action stops before generating DUMP data sets.

User response:
Follow your local site practices to grant the user READ access to the directory. WRITE access is also recommended as it will be used during the GENERATE action too.

KFJ00209E Directory does not exist: *name*

Explanation:
z/OS UNIX System Services directory does not exist. Specified directory should be created during GENERATE action.

System action:
The **PACKAGE** action stops before generating DUMP data sets.

User response:

Review GENERATE JCL Job output log for any z/OS UNIX errors.

KFJ00210E Unable to create PAX archive *name*

Explanation:
Unexpected error while trying to create PAX archive on configuration system.

System action:
The **PACKAGE** action stops before generating DUMP data sets.

User response:
Verify job output for more details. Consider multiple issues: permissions, space shortage.

KFJ00211I Configuration Manager is allocating data sets using local settings for high-level qualifiers and z/OS Unix system service paths

Explanation:
During RTE generation, KFJ_LOCAL_* local parameters will be used to allocate runtime environment data sets instead of RTE_* or GBL_* parameters.

System action:
The **GENERATE** action performs Run Time Environment generation using local parameters.

User response:
Review mapping table and ensure that all of the KFJ_LOCAL_* parameters are specified correctly.

KFJ00212E No DUMP data sets found to deploy

Explanation:
The **DEPLOY** action did not find any DUMP data sets to restore.

System action:
The **DEPLOY** action stops with a return code of 8.

User response:
Check that expected DUMP data sets are available on the system. Verify KFJ_PACK_* parameters, if they are used.

KFJ00213E Customizing parameter *parameter_name* is not supported by the GENERATE action when using option parameter KFJ_LOCAL_PLIB_HILEV. Use the product provided default.

Explanation

The **KFJ_LOCAL_PLIB_HILEV** parameter was specified in the JCL (KCIVARS DD statement). The **GENERATE** action does not allow certain parameter customization for runtime environment generation for remote systems.

In the following example, one of the parameters, **GBL_USER_JCL**, was found in the RTEDEF members, which caused the program to stop with a return code of 8:

```
KFU00002I INVOKE processing is about to
commence; MEMBER=KFJOMEGA DDNAME=KCIFLOW
DSN=TEST1.SYSPLEX.SMPE.TKANCUS Maintenance
level: OA61601
01. Using parameters in
TEST1.CM.R683L.RTEDEF(PGVK)
02. Using parameters in
TEST1.CM.R683L.RTEDEF(KDS$PARM)
03. Using parameters in
TEST1.CM.R683L.RTEDEF(GBL$PARM)
04. Using parameters in
TEST1.CM.R683L.RTEDEF(PCK$PARM)
KFJ00213E Customizing parameter GBL_USER_JCL is
not supported by the GENERATE action when
 using option parameter KFJ_LOCAL_PLIB_HILEV.
Use the product provided default.
KFU00086E REXX routine has failed;
EXEC=VALVARS  RC=8

KFU00004I KCIOMEGA is ending; RC=8
SYSPLEX=RSPLEX0K LPAR=1234 DATE=2021-08-25
08:23:21 Thursday
```

System action:
The **GENERATE** action stops with return code 8.

User response

Review parameter *parameter_name* to ensure that it is not used by the **GENERATE** action. To achieve this, you can comment out this parameter from the RTEDEF members and restart the **GENERATE** action.

If parameter *parameter_name* is related to KD2 operational data set allocation, you can resolve this error by using parameter **KFJ_LOCAL_KD5_RUN_ALLOC** set to value **DEPLOY** or **NONE**.

KFJ00214E Unable to extract PAX archive

Explanation:
An unexpected error occurred while trying to extract the PAX archive on a remote (target) system.

System action:
The **DEPLOY** action stops with a return code of 8.

User response:
Review the job output for more details. Consider multiple issues: permissions, space shortage, corrupted data set.

KFJ00215W RTE_X_HILEV_SHARING not supported. Parameter ignored.

Explanation:
Monitoring Configuration Manager does not support the **RTE_X_HILEV_SHARING** parameter. The parameter value does not migrate from the existing

PARMGEN configuration to the Configuration Manager RTEDEF data set.

System action:
The **RTE_X_HILEV_SHARING** parameter value is ignored and not used. Processing continues.

User response:
No action is required.

KFJ00216E OPTION *option* not supported

Explanation:
The **OPTION** parameter value specified in KCIVARS is not supported.

System action:
The action stops with a return code of 8.

User response:
Review the **OPTION** parameter value specified. Make sure that the option value is spelled correctly and that is available for the specific action.

KFJ00217E OPTION *option_1* not compatible with option *option_2*

Explanation:
Certain options are not compatible due to conflicting outcomes.

System action:
The action stops with a return code of 8.

User response:
Remove one or more of the conflicting options and rerun the job.

KFJ00218E *$PARM and/or *$GLOB members already exist

Explanation:
The **MIGRATE** action checked the existing *rte_plib_hilev*.RTEDEF and found sysplex level members in the library.

System action:
The **MIGRATE** action stops with a return code of 8.

User response:
Delete or rename the members listed in KCIPRINT and rerun the **MIGRATE** action.

KFJ00219E *$lpar members already exist

Explanation:
The **MIGRATE** action checked the existing *rte_plib_hilev*.RTEDEF and found LPAR-specific members in the library.

System action:
The **MIGRATE** action stops with a return code of 8.

User response:
Delete or rename the members listed in KCIPRINT and rerun the **MIGRATE** action.

KFJ00220E RTE_NAME member already exists

Explanation:
The **MIGRATE** action checked the existing *rte_plib_hilev*.RTEDEF and found the RTE_NAME member in the library.

System action:
The **MIGRATE** action stops with a return code of 8.

User response:
Delete or rename the RTE_NAME member listed and rerun the **MIGRATE** action.

KFJ00221I DEBUG does not impact the deletion of DDs in JES3

Explanation:
JES3 does not support the deletion of DD outputs from the spool, so the function of **OPTION DEBUG** will not have impact on this function.

System action:
No impact. Processing continues.

User response:
No action is required.

KFJ00222I Package *data_set_name* not found

Explanation:
The **DEPLOY** action expects certain data sets to be available during the restore process. The availability of the data sets depends on the results of the **PACKAGE** action.

System action:
No impact. Processing continues.

User response:
Review the results from the **PACKAGE** action. Ensure that all data sets created by the **PACKAGE** action are processed during the **DEPLOY** action.

KFJ00223I Global RTE PDS V1 setting is ON | OFF

Explanation

This message denotes the global persistent data store version 1 status for the runtime environment. This status is the final status that is set by the configuration process and is also written to the KPQHINIT member in RKANPARU. This status applies to all agents and does not reflect the initial status of, for example, **RTE_PDS2_ACTIVATION**, or individual **Kpp_PDS2_ACTIVATION** flags. This value is derived from the combined parameter flags that are used in the runtime environment.

This message is always followed by message KFJ00224I. If PDS V1 is ON, details about why PDS V1 is ON appear after message KFJ00224I. This information includes a list of agents for which PDS V2

is not activated and also indicates if PDS V2 is not activated due to an unsupported agent version. For more information about supported product versions, see PDS V2 support.

For more information, see How to: Activate PDS V2.

System action:
No impact. Processing continues.

User response:
No action is required.

KFJ00224I Global RTE PDS V2 setting is ON | OFF

Explanation

This message denotes the global persistent data store version 2 status for the runtime environment. This status is the final status that is set by the configuration process, which is also written to the KPQHINIT member in RKANPARU. This status applies to all agents and does not reflect the initial status of, for example, **RTE_PDS2_ACTIVATION**, or individual **Kpp_PDS2_ACTIVATION** flags. This value is derived from the combined parameter flags that are used in the runtime environment.

For more information, see How to: Activate PDS V2.

System action:
No impact. Processing continues.

User response:
No action is required.

KFJ00225E EMBEDS data set invalid or not allocated - *rte_x_override_embeds_lib*

Explanation:
If parameter **RTE_X_OVERRIDE_EMBEDS_LIB** is found in the RTE_NAME member of RTEDEF, it must be set to a valid value that corresponds to an existing embeds library. If the parameter value is empty or mistyped, this error is issued. The message provides the specified parameter value in *rte_x_override_embeds_lib*; an empty string indicates that the parameter value was empty.

System action:
Processing stops.

User response:
Either correct the value specified in parameter **RTE_X_OVERRIDE_EMBEDS_LIB**, or, if no embed overrides are required, remove the parameter from the RTE_NAME member.

KFJ00226W Non-secure TEMS communication protocol is used

Explanation:
Runtime environment parameter **RTE_TEMS_TRANSPORT_MODE** is set to HTTP. The runtime environment is using the non-secure HTTP communication protocol for all agents.

System action:
Processing continues.

User response:
Consider changing the communication protocol to HTTPS. For more information, see Update runtime environment to use HTTPS.

KFJ00227E *hilev_parameter* parameter value must match a data set name used in JCL KCIFLOW DD statement

Explanation

The high-level qualifier value for the SMP/E target libraries that was specified in parameter *hilev_parameter* was not found in the KCIFLOW DD statement in the JCL, where *hilev_parameter* is **GBL_TARGET_HILEV** or **KFJ_LOCAL_TARGET_HILEV**. The KCIFLOW DD statement and the RTEDEF members must point to the same SMP/E target libraries.

For the **GBL_TARGET_HILEV** parameter, the parameter and value are specified in the GBL$PARM or GBL$*lpar* member.

For the **KFJ_LOCAL_TARGET_HILEV** parameter, the parameter and value are specified in the PCK$PARM or PCK$*lpar* member.

For example, in the GBL$PARM or GBL$*lpar* member, the high-level qualifier for the SMP/E target libraries is defined as follows:

```
GBL_TARGET_HILEV     "MONSUITE"
```

In the JCL, this same high-level qualifier value must be specified in the KCIFLOW DD statement, as shown:

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ZOS1
//S1       EXEC PGM=KCIOMEGA,REGION=0M,DYNAMNBR=256
//STEPLIB  DD DISP=SHR,DSN=MONSUITE.TKANMOD
//KCIFLOW  DD DISP=SHR,DSN=MONSUITE.TKANCUS(KFJOMEGA)
//KCIVARS  DD *
ACTION            GENERATE
RTE_NAME          RTE1
RTE_PLIB_HILEV    TSOUID.MONSUITE
```

System action:
Processing stops.

User response

Ensure that the KCIFLOW DD statement in the JCL points to the same SMP/E target libraries that are used inside the RTEDEF members.

- For a standard **GENERATE** action, verify the **GBL_TARGET_HILEV** parameter value.

- If **KFJ_LOCAL_PLIB_HILEV** is specified in the KCIVARS DD statement, verify the **KFJ_LOCAL_TARGET_HILEV** parameter value.

KFJ00228W **RTE_SHARE *xxx* not supported. RTE_SHARE changed to SMP.**

Explanation:
Configuration Manager supports RTE_TYPE FULL, or RTE_TYPE SHARING with RTE_SHARE SMP. However, the **MIGRATE** action has detected that the PARMGEN WCONFIG configuration uses an RTE_SHARE value that is not equal to SMP. If RTE_TYPE is set to SHARING, and RTE_SHARE is not equal to SMP, you will receive this warning message.

System action:
RTE_SHARE is reset to the default value, which is SMP.

User response:
None. If you were using a base library with PARMGEN, you might consider using the Configuration Manager target copy feature, which allows you to create and use a copy of your SMP/E target libraries. For more information, see "Using SMP/E target library copies" on page 141.

KFJ00229E **Failed to copy files from *source_path* to TRG_COPY_TKANJAR_PATH directory *target_path***

Explanation:
The **GENERATE** action failed to copy files from the directory specified in parameter **GBL_USS_TKANJAR_PATH** to the directory specified in **TRG_COPY_TKANJAR_PATH**.

System action:
The **GENERATE** action stops before copying installation files.

User response:
Ensure that the directory specified in parameter **GBL_USS_TKANJAR_PATH** contains all the required files. Ensure that the directory specified in parameter **TRG_COPY_TKANJAR_PATH** exists and the user who runs the job that performs the **GENERATE** action has the required permissions to create new files.

KFU messages

Messages with the prefix KFU are from the KCIOMEGA program or its APF-authorized version, KCIALPHA.

KCIOMEGA is the underlying job template engine that runs Monitoring Configuration Manager.

KFU00001I **KCIOMEGA is starting; SYSPLEX=*name* LPAR=*name* DATE=*date and time***

Explanation:
The KCIOMEGA workflow utility is starting. The system, date and time are reported. The KCIALPHA utility, the APF-authorized version of KCIOMEGA, issues the same message.

System action:
Processing continues.

User response:
None required.

KFU00002I ***command* processing is about to commence; MEMBER=*name* DDNAME=*name* DSN=*name***

Explanation:
A command to process a sub-workflow (**INVOKE**) or read parameters (**CONFIG**) is about to commence. The source member, ddname, and data set name are reported.

System action:
For **INVOKE**, the sub-workflow is given control, returning the invoking workflow upon completion. For **CONFIG**, the parameters are read in and made available for the workflow to reference.

User response:
None required.

KFU00003I **Workflow task recap: Programs=*program_count* MaxRC= *max_program_rc* REXX=*rexx_exec_count* MaxRC=*max_rexx_rc***

Explanation:
Workflow processing has ended. The number of programs and REXX execs that were invoked is reported, along with the maximum return code for each.

System action:
Processing ends.

User response:
If either return code is greater than zero, then look in the KCIPRINT sysout data set for error or warning messages that indicate the cause of the problem and recommend corrective action. The return code issued in message KFU00004I will help determine if the return codes issued in KFU00003I are acceptable. If you suspect an error, contact IBM Software Support.

KFU00004I **KCIOMEGA is ending; RC=*return_code* SYSPLEX=*sysplex_name* LPAR=*LPAR_name* DATE=*date_and_time***

Explanation

The KCIOMEGA workflow utility is ending. The reported return code, *return_code*, represents the maximum return code for the Configuration Manager action that was performed. The message also provides the system information, date, and time.

Return code	Description
0	The workflow completed with no errors.
4	The workflow completed with one or more warnings. To determine the significance of the warnings, review the preceding messages in the KCIPRINT sysout data set and also, if necessary, KCITRACE. **Tip:** The **DISCOVER** action ends with return code 4 if you are performing *rediscovery*: if you have previously performed discovery for an LPAR, and RTEDEF(K*pp*@*lpar*) members already exist. Instead of overwriting those members, the **DISCOVER** action writes RTEDEF(K*pp*#*lpar*) members. For details, see "Members created by the DISCOVER action" on page 41.
8	The workflow stopped processing due to an unrecoverable error.

The workflow can also set its own return code via the **STOP** command.

System action:
Processing ends.

User response:
If the return code is greater than zero, then a previous error or warning message indicates the cause of the problem and recommends corrective action.

KFU00005I **Program is about to be invoked; PROGRAM=*name***

Explanation:
The program is about to be invoked to perform a task.

System action:
The program is given control, typically as a subtask.

User response:
None required.

KFU00006I **Program has ended; PROGRAM=*name* OUTPUT=*output* WORKFLOW=*workflow* STEP=step RC=*return_code***

Explanation:
The program has ended with the reported return code. The program is deemed to have succeeded because the return code is expected. Specifically, the return code is less than the MAXRC setting of the active workflow.

System action:
Processing continues.

User response:
No action is required.

KFU00007E **Program has abnormally terminated; PROGRAM=*name* ABEND=*system code***

Explanation:
The program has abnormally terminated with the reported system code.

System action:
Processing of the workflow stops.

User response:
z/OS MVS System Codes in IBM Documentation describes the abend system code. Additional diagnostic messages may be recorded in the job log. The abend system code or job log messages may recommend or indicate corrective action. Otherwise, re-run the job with a SYSUDUMP DD to generate a dump and report the problem to IBM Software Support.

KFU00008I **Workflow was told to stop**

Explanation:
The **STOP** command was issued by the workflow, typically as a result of an error condition, incorrect parameter input, or further action required.

System action:
Processing of the workflow stops.

User response:
See the KCIPRINT sysout data set for a message that describes why the workflow stopped, and then perform the recommended action.

KFU00009E **Workflow has stopped due to an unrecoverable error**

Explanation:
An unrecoverable error occurred in the workflow.

System action:
Processing of the workflow stops.

User response:

A previous error or warning message will indicate the cause of the problem and recommend corrective action.

KFU00010E DD statement is missing, DDNAME=KCIPRINT or KCITRACE

Explanation:
The reported file, either KCIPRINT or KCITRACE, is not allocated. If it is not explicitly specified in the job's JCL as a DD statement, then it is dynamically allocated to SYSOUT=*. Therefore, this error should not occur in normal circumstances and indicates an environmental problem that KCIOMEGA cannot recover from.

System action:
Processing stops.

User response:
Check in the job log for messages associated with the ddname and take corrective action. Otherwise contact IBM Software Support.

KFU00011E DD statement is missing, DDNAME=*name*

Explanation:
The reported file, typically a system-generated ddname, is not allocated. The file was dynamically allocated but could not be found. This error should not occur in normal circumstances and indicates an environmental problem that KCIOMEGA cannot recover from.

System action:
Processing stops.

User response:
Check in the job log for messages associated with the ddname and take corrective action. Otherwise contact IBM Software Support.

KFU00012E MVS service has failed; MACRO=BLDL PROGRAM=*name* R15=*return code* R0=*reason code*

Explanation:
The reported program, about to be run by the workflow, could not be located.

System action:
Processing stops.

User response:
For return code 4, check that the program exists in the workflow STEPLIB. For all other return codes, refer to the **BLDL** completion codes in IBM Documentation for corrective action. Otherwise, contact IBM Software Support.

KFU00013E MVS service has failed; MACRO=LOAD PROGRAM=*name* R1=*system code* R15=*reason code*

Explanation:
The reported program, about to be run by the workflow, could not be loaded.

System action:
Processing stops.

User response:
Check the job log for messages associated with the failed load request. A common abend condition is S806-04 indicating that the module could not be found. Check that the program exists in the workflow STEPLIB. Otherwise, contact IBM Software Support.

KFU00014E MVS service has failed; MACRO=LINK PROGRAM=*name* R1=*system code* R15=*reason code*

Explanation:
The reported program, about to be run by the workflow, could not be invoked.

System action:
Processing stops.

User response:
Check the job log for messages associated with the failed link request. A typical abend condition is S806-04 indicating that the module could not be found. Check that the program exists in the workflow STEPLIB. Otherwise, contact IBM Software Support.

KFU00015E MVS service has failed; MACRO=ATTACH PROGRAM=*name* R15=*return code*

Explanation:
The reported program, about to be run by the workflow, could not be invoked.

System action:
Processing stops.

User response:
Check the job log for messages associated with the failed attach request. Refer to the **ATTACH** return codes in IBM Documentation for corrective action. Otherwise, contact IBM Software Support.

KFU00016E Program has failed; PROGRAM=*name* OUTPUT=*output* WORKFLOW=*workflow* STEP=*step* RC=*return_code*

Explanation:
The program invoked by the workflow has ended with the reported return code. The program is deemed to have failed because the return code is higher than expected. Specifically, the return code is not less than the MAXRC setting of the active workflow.

System action:
Processing stops.

User response

The failing program will typically issue its own error messages to indicate the cause of the problem and recommend corrective action. These messages can be written to the following locations:

- In the KCIPRINT or KCITRACE sysout data set, immediately prior to this message.
- The output data set associated with the program or utility. OUTPUT=*output* in the error message identifies this data set.
- The job log containing the system messages for the job.

IBM Z Monitoring Configuration Manager uses system and OMEGAMON utilities to configure the runtime environment. These utilities control their own output messages; those messages will typically not appear in KCIPRINT or KCITRACE.

Utilities such as IEBGENER, IEBCOPY, and IDCAMS write messages to their SYSPRINT output data set. Note that the KCIOMEGA program might have renamed the SYSPRINT ddname to the name of the workflow step that invoked the program.

The OMEGAMON utility KCIPARSE is used extensively to prepare the runtime members. Error messages might be written to either the associated SYSPRINT or to the job log.

If you cannot locate an associated error message or that message does not recommend corrective action, then contact IBM Software Support.

KFU00019E **Variable value end quote is missing; VAR=*name source***

Explanation:
The reported variable or parameter has an invalid value. The value is assumed to be enclosed in quotes because it starts with a quote, but the end quote is missing. Additional information is recorded in the message to identify the source of the variable, typically the KCIVARS data set or an RTEDEF library member.

System action:
Processing stops.

User response

1. Ensure that the variable value, if it contains embedded blanks, is enclosed in quotes.
2. Ensure that the variable value, including quotes, does not extend beyond column 70.
3. Retry the request.

KFU00020E **CONFIG or INVOKE MEMBER= command has exceeded the maximum nesting level**

Explanation:
The **CONFIG** or **INVOKE** command in the workflow could not be run because it will exceed the maximum level of command nesting permitted.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00021E **Command is invalid: *workflow statement***

Explanation:
The reported workflow statement has invalid syntax.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00022E **INVOKE member was not found; Member=*name* DSN=*name***

Explanation:
The workflow **INVOKE** command could not find the member to be invoked in the reported data set.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00023W **CONFIG member was not found; Member=*name* DSN=*name***

Explanation:
The workflow **CONFIG** command could not find the member to be processed in the reported data set.

System action:
Processing continues, without any new parameters.

User response:
If the workflow expects and requires the **CONFIG** member then create the member and re-run the job. If the **CONFIG** is intended to provide optional parameters only then no action is required.

KFU00024E **RC is not in the range 0 to 2147483647: *return code***

Explanation:
The workflow tried to set the return code variable (RC) with a value outside the allowed range.

System action:
Processing stops.

User response:

If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00025E DD statement is missing, DDNAME=*name* RC=*return code*

Explanation:
The reported file, typically a system-generated ddname, is not allocated. The file was dynamically allocated but could not be found. This error should not occur in normal circumstances and indicates an environmental problem that KCIOMEGA cannot recover from.

System action:
Processing stops.

User response:
Check the job log for messages associated with the ddname and take corrective action. Otherwise contact IBM Software Support.

KFU00026E DD OPEN error, DDNAME=*name* ABEND=*system code-reason code*

Explanation:
The reported file, typically a system-generated ddname, could not be opened. A common abend condition is S913 indicating that access to the data set is not allowed by the security server, such as RACF.

System action:
Processing stops.

User response:
Check the job log for messages associated with the ddname and take corrective action. Otherwise contact IBM Software Support.

KFU00027E Copy I/O error, DDNAME=*name* ABEND=*system code-reason code*

Explanation:
The request to copy data from one file to another has failed.

System action:
Processing stops.

User response:
Check the job log for messages associated with the ddname and take corrective action. Otherwise contact IBM Software Support.

KFU00030E Workflow command is not recognized; Member=*name* Line=*number*

Explanation:
The workflow encountered a command that was not recognized. The workflow member name and line number within the workflow identify the offending command.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00031E JCL statement has a syntax error; Member=*name* Line=*number*

Explanation:
The workflow encountered a JCL statement that was not recognized. The workflow member name and line number within the workflow identify the offending statement.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00032E Workflow command is invalid, Member=*name* Line=*number*

Explanation:
The workflow encountered a command with invalid syntax or used out of context. The workflow member name and line number within the workflow identify the offending command.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00037E String in quotes is not terminated; Member=*name* Line=*number*

Explanation:
The workflow encountered a string that started with a quote but was not terminated with a quote. The workflow member name and line number within the workflow identify the offending command.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00040E JCL statement name is longer than 8; *statement*

Explanation:
The reported JCL statement has a name longer than 8 characters. For an EXEC statement this is the step name. For a DD statement this is the ddname.

System action:
Processing stops.

User response:

If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00041E JCL statement operation is not EXEC or DD; *statement*

Explanation:
The reported JCL statement does not specify a recognized operation. Only EXEC and DD statements are supported.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00042E DD statement has an unsupported keyword parameter value; *parameter*

Explanation:
The reported JCL DD statement has specified a keyword parameter with an unsupported value. Only some of the actual JCL DD statement parameter values are supported in workflows.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00043E DD statement keyword parameter value is missing; *parameter*

Explanation:
The reported JCL DD statement has specified a keyword parameter that has no value. This typically occurs when the value is parameterized, and the parameter is not defined or has no value. Some parameters support missing or null values, in which case the system default is used. Other keyword parameters, such as **DSNAME**, if specified in the DD statement must be resolved to an allowed value.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00044W DD statement keyword parameter is not recognized; *statement*

Explanation:
The reported JCL DD statement has specified a keyword parameter that is not supported in workflows. Only some of the actual JCL DD statement parameters are supported in workflows.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00045E EXEC statement has an unsupported keyword parameter value; *statement*

Explanation:
The reported JCL EXEC statement has specified a keyword parameter with an unsupported value. Only some of the actual JCL EXEC statement parameter values are supported in workflows.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00046E EXEC statement keyword parameter value is missing; *statement*

Explanation:
The reported JCL EXEC statement has specified a keyword parameter that has no value. This typically occurs when the value is parameterized, and the parameter is not defined or has no value. Some parameters support missing or null values, in which case the system default is used. Other keyword parameters, such as **PGM**, must be resolved to an allowed value.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00047W EXEC statement keyword parameter is not recognized; *statement*

Explanation:
The reported JCL EXEC statement has specified a keyword parameter that is not supported in workflows. Only some of the actual JCL EXEC statement parameters are supported in workflows.

System action:
Processing stops.

User response:
If the problem occurs in an IBM-supplied workflow then contact IBM Software Support.

KFU00050E DYNALLOC request has failed; REQUEST=ALLOC EC=*error code* IC=*information code* DSN=*data set name*

Explanation:
The DD statement in the workflow failed dynamic allocation with the reported error code. The message severity is reduced to a warning if the dynamic allocation is used only to check for the existence of the data set. Additional messages issued by dynamic allocation will be reported immediately after this message, explaining the problem.

System action:
Processing stops when the message severity is an error. Processing continues when the message severity is a warning.

User response:
Refer to the additional messages issued by dynamic allocation for corrective action. If the problem persists then contact IBM Software Support.

KFU00051I **DYNALLOC request has failed; REQUEST=ALLOC EC=*error code* IC=*information code* DSN=*data set name***

Explanation

The DD statement in the workflow failed dynamic allocation with the reported error code.

The dynamic allocation error is informational because the allocation is used only to check for the existence of the data set, or to allocate it. The `WARNING=RETURN` option was specified in the DD statement.

See message KFU00050E for more information about dynamic allocation errors.

System action:
Processing continues.

User response:
None required.

KFU00055E **MVS service has failed; MACRO=$SWAREQ R15=*return code* SVA=*address* CB=*control block name***

Explanation:
An internal service similar to SWAREQ was invoked to extract an SWA control and has failed.

System action:
Processing stops.

User response:
If the problem persists then contact IBM Software Support.

KFU00060E **DD statement is missing, DDNAME=*name* RC=*return code***

Explanation:
The reported file, typically a system-generated ddname, is not allocated. The file was dynamically allocated but could not be found. This error should not occur in normal circumstances and indicates an environmental problem that KCIOMEGA cannot recover from.

System action:
Processing stops.

User response:
Check the job log for messages associated with the ddname and take corrective action. Otherwise contact IBM Software Support.

KFU00061E **MVS service has failed; MACRO=DESERV FUNC=GET R15=*return code* R0=*reason code* DDNAME=*name* DSN=*name***

Explanation:
The DESERV system service was invoked to provide information about a member in a data set, but failed with the reported return code.

System action:
Processing stops.

User response:
Check the job log for system messages associated with this service and take corrective action. Otherwise contact IBM Software Support.

KFU00062I **PDS library data set is empty; MACRO=DESERV FUNC=GET_ALL R15=*return code* R0=*reason code* DDNAME=*name* DSN=*name***

Explanation:
The DESERV system service was invoked to provide the list of members in a data set, but the data set is empty.

System action:
Processing continues.

User response:
None required.

KFU00063E **MVS service has failed; MACRO=DESERV FUNC=GET_ALL R15=*return code* R0=*reason code* DDNAME=*name* DSN=*name***

Explanation:
The DESERV system service was invoked to provide the list of members in a data set, but failed with the reported return code.

System action:
Processing stops.

User response:
Check the job log for system messages associated with this service and take corrective action. Otherwise contact IBM Software Support.

KFU00064E **REXX service has failed; Routine=IRXEXCOM RETC=*return code* VAR=*name***

Explanation:
The IRXEXCOM service was invoked to set the value of a REXX variable but failed with the reported return code.

System action:
The REXX exec will throw an exception that either causes the exec to fail or give unpredictable results.

User response:
Contact IBM Software Support.

KFU00070E **DD statement is missing, DDNAME=*name* RC=*return code***

Explanation:
The reported file, typically a system-generated ddname, is not allocated. The file was dynamically allocated but could not be found. This error should not occur in normal circumstances and indicates an environmental problem that KCIOMEGA cannot recover from.

System action:
Processing stops.

User response:
Check the job log for messages associated with the ddname and take corrective action. Otherwise contact IBM Software Support.

KFU00071E **MVS service has failed; MACRO=DESERV FUNC=DELETE R15=*return code* R0=*reason code* DDNAME=*name* DSN=*name***

Explanation:
The DESERV system service was invoked to delete a member in a data set, but failed with the reported return code.

System action:
Processing stops.

User response:
Check the job log for system messages associated with this service and take corrective action. Otherwise contact IBM Software Support.

KFU00072E **MVS service has failed; MACRO=STOW FUNC=DELETE R15=12 R0=1234 DDNAME=12345678 DSN=**

Explanation:
The STOW system service was invoked to delete a member in a data set, but failed with the reported return code.

System action:
Processing stops.

User response:
Check the job log for system messages associated with this service and take corrective action. Otherwise contact IBM Software Support.

KFU00080E **REXX service has failed; Routine=IRXINIT RETC=*return code* REAS=*reason code***

Explanation:
The IRXINIT service was invoked to initialize the REXX environment but failed with the reported return code.

System action:
Processing stops.

User response:
Contact IBM Software Support.

KFU00081I **REXX routine is about to be invoked; EXEC=*name***

Explanation:
The REXX exec is about to be invoked to perform a task.

System action:
The exec is given control under the control of the REXX environment.

User response:
None required.

KFU00082I **REXX routine has completed; EXEC=*name* OUTPUT=*output* WORKFLOW=*workflow* STEP=*step* RC=*return_code***

Explanation:
The REXX exec has ended with the reported return code. The REXX exec is deemed to have succeeded because the return code is expected. Specifically, the return code is less than the MAXRC setting of the active workflow.

System action:
Processing continues.

User response:
No action is required.

KFU00083E **REXX service has failed; Routine=IRXTERM RETC=*return code***

Explanation:
The IRXTERM service was invoked to terminate the REXX environment but failed with the reported return code.

System action:
Processing stops.

User response:

Contact IBM Software Support.

KFU00084E **REXX service has failed; Routine=IRXEXCOM RETC=*return code* VAR=*name***

Explanation:
The IRXEXCOM service was invoked to set the value of a REXX variable but failed with the reported return code. The service was invoked during REXX initialization to populate the REXX variable pool with the workflow variables.

System action:
The REXX exec will be processed, but some of the workflow variables will not be available to the REXX exec.

User response:
Contact IBM Software Support.

KFU00085E **REXX service has failed; Routine=IRXSAY RETC=*return code***

Explanation:
The IRXSAY service was invoked to write a message to the REXX output file but failed with the reported return code.

System action:
The message is not issued and the REXX exec processing continues. The message was likely written to the KCITRACE sysout data set as a backup.

User response:
Contact IBM Software Support.

KFU00086E **REXX routine has failed; EXEC=*name* OUTPUT=*output* WORKFLOW=*workflow* STEP=step RC=*return_code***

Explanation:
The REXX exec invoked by the workflow has ended with the reported return code. The REXX exec is deemed to have failed because the return code is higher than expected. Specifically, the return code is not less than the MAXRC setting of the active workflow.

System action:
Processing stops.

User response

The failing REXX will typically issue its own error messages to indicate the cause of the problem and recommend corrective action. These messages can be written to the following locations:

- In the KCIPRINT or KCITRACE sysout data set, immediately prior to this message.
- The output data set associated with the REXX exec. OUTPUT=*output* in the error message identifies this data set.
- The job log containing the system messages for the job.

REXX typically writes messages to the SYSTSPRT output data set. The KCIOMEGA program might have renamed the SYSTSPRT ddname to the name of the workflow step that invoked the REXX.

If you cannot locate an associated error message or that message does not recommend corrective action, then contact IBM Software Support.

KFU00090E **Command is not recognized; REXX=*name* COMMAND=*command***

Explanation:
The KCIEXEC host command environment set up for REXX processing did not recognize the command in the exec.

System action:
The REXX exec with throw a failure that will either stop exec processing or cause unpredictable results.

User response:
If the REXX is supplied by IBM then contact IBM Software Support.

KFU00091E **Command is not recognized; REXX1=*name* REXX2=*name* COMMAND=*command***

Explanation:
This message is a variation of message KFU00090E. In this case, the REXX exec that issued the command is not the original EXEC REXX=*name* specified in the workflow. REXX1 is the original workflow exec and REXX2 is the most recently called exec.

System action:
The REXX exec with throw a failure that will either stop exec processing or cause unpredictable results.

User response:
If the REXX is supplied by IBM then contact IBM Software Support.

KFU00092W **Command is not supported; REXX=*return code* COMMAND=*command***

Explanation:
The KCIEXEC host command environment set up for REXX processing recognized the command but does not support it. For example, the **SUBMIT** command is recognized but is not performed.

System action:
The REXX exec treats the command as a null operation and continues.

User response:
If the task needs to be performed then you must do that manually after the workflow has completed. For example, you can submit the job after the workflow has completed.

KFU00093W **Command is not supported; REXX1=*name* REXX2=*name* COMMAND=*command***

Explanation:
This message is a variation of message KFU00092W. In this case, the REXX exec that issued the command is not the original EXEC REXX=*name* specified in the workflow. REXX1 is the original workflow exec and REXX2 is the most recently called exec.

System action:
The REXX exec treats the command as a null operation and continues.

User response:
If the task needs to be performed then you must do that manually after the workflow has completed. For example, you can submit the job after the workflow has completed.

KFU00095E **REXX service has failed; Routine=IRXEXCOM RETC=*return code***

Explanation:
The IRXEXCOM service was invoked to fetch or store one or more REXX variables but failed with the reported return code. The service was invoked during the processing of a KCIEXEC request.

System action:
The REXX exec will continue processing but results in exec or workflow will be unpredictable.

User response:
If the REXX is supplied by IBM then contact IBM Software Support.

KFU00100W **Full discovery requires APF authorization; PROGRAM=*name***

Explanation:
The subsystem discovery service was invoked using a program that is not APF-authorized. KCIALPHA is the APF-authorized version of KCIOMEGA and is recommended for discovery. If KCIALPHA is the program that was used, then the job's STEPLIB data set is not an APF-authorized library.

System action:
Discovery continues, all the subsystems will be discovered, but some of their attributes will be incomplete.

User response

For a compete discovery, APF-authorize the TKANMOD library using the following system command:

```
SETPROG APF,ADD,DSNAME=hlq.TKANMOD,SMS
```

Otherwise manually edit the discovery members in the RTEDEF library to complete the process.

KFU00101E **Discovery has abended; ABEND=12345678 SSID=*name* PHASE=*diagnostics***

Explanation:
The subsystem discovery service has abended during the analysis of the reported subsystem. If the subsystem is not Db2, MQ, CICS, or IMS then this is not a problem. This problem might occur when discovery is scanning control blocks to detect the type of subsystem, but encounters a control block whose storage is inaccessible.

System action:
Discovery ignores this subsystem and moves onto the next subsystem.

User response:
If the subsystem is Db2, MQ, CICS or IMS then contact IBM Software Support. Otherwise the message can be ignored.

KFU00102E **MVS service has failed; MACRO=ALESERV FUNCTION=ADD RC=*return code***

Explanation:
Discovery is using the ALESERV system service to access, via cross-memory, the subsystem address space in order to extract its discoverable information but the service has failed with the reported return code.

System action:
Discovery continues, the subsystem will be discovered, but some of its attributes will be incomplete.

User response:
If the problem persists then contact IBM Software Support.

KFU00103E **MVS service has failed; MACRO=ALESERV FUNCTION=DELETE RC=*return code***

Explanation:
Discovery is using the ALESERV system service to terminate its access to the subsystem address space but the service has failed with the reported return code.

System action:

Discovery continues, the subsystem is discovered, and its attributes will be incomplete.

User response:
If the problem persists then contact IBM Software Support.

KFU00104E **MVS service has failed; MACRO=LOAD PROGRAM=EZBNMIFR R1=*<abend_code>* R15=*<reason_code>***

Explanation:
Network discovery could not load program EZBNMIFR, which is the TCP/IP NMI service. A common problem is the ABENDS806-04 module is not found. Typically, EZBNMIFR is in SYS1.CSSLIB, the common services library, which is always included in the LINKLIST.

System action:
The TCPIP stacks are discovered with incomplete information.

User response:
Copy module EZBNMIFR to the LINKLIST and retry the request. To proceed with the configuration process, first review member KN3@<*lpar*> in your RTEDEF library and correct the parameters.

KFU00105E **NMI (EZBNMIFR) failed to get TN3270 settings; RET=*<return_code>* REAS=*<reason_code>* TCPIP=*<name>***

Explanation

Network discovery issued a TCP/IP NMI service poll request to obtain the TN3270 servers with an affinity to the reported TCPIP stack. The request failed with the reported return and reason codes. When RET=0000006F, permission is denied (EACCES) because the discovery job is not APF authorized. Message KFU00100W is issued prior to this message and recommends corrective action.

System action:
The TCPIP stacks are discovered with incomplete information.

User response:
If the return code indicates a program error then contact IBM. To proceed with the configuration process, first review member KN3@<*lpar*> in your RTEDEF library and correct the parameters.

KFU00110I **Some SYSOUT files were not deleted; count=*number***

Explanation:
Some of the output files that are not important could not be deleted. The number of files not deleted is reported. The files will remain part of the output for the completed job.

System action:
This situation does not impact the function being performed by the workflow and has no effect on the final return code of the job. Processing continues.

User response:
If the situation occurs in an IBM-supplied workflow, then contact IBM Software Support.

KFU00111I **SYSOUT file delete was requested; DDNAME=*name* DSN=*name***

Explanation:
The output file is not important, and its delete request was sent to JES2. In certain situations, the request may not be successful. This occurrence is normal and can occur when the output is busy being processed by JES2.

System action:
Processing continues. If the output was not deleted on the first attempt, then the request will be retried later.

User response:
No action is required.

KFU00112I **SYSOUT file is not used; DDNAME=*name* DSN=*name***

Explanation:
The output file is eligible to be deleted but could not be found. It is likely that the file is already deleted because it was not used. You might consider removing it from the workflow.

System action:
Processing continues.

User response:
No action is required.

KFU00113I **SYSOUT file could not be located; RDJFCB RC=*return_code* DDNAME=*name***

Explanation:
The output file is eligible to be deleted because it is not important. However, it will not be deleted because its JES data set name could not be located. The file will remain part of the output for the completed job.

System action:
This situation does not impact the function being performed by the workflow and has no effect on the final return code of the job. Processing continues.

User response:
If the situation occurs in an IBM-supplied workflow, then contact IBM Software Support.

KFU00114I **JES STATUS request failed: FUNCTION=SSST(80)**

STATTYPE=*code* RC=*return_code*
SSOBRETN=*return_code*
STATREAS=*reason_code*
JOBNAME=*name* JOBID=*id*

Explanation:
The JES subsystem interface Extended status function call (SSI 80) was made to locate the output files created by the workflow steps. The request failed with the reported return and reason codes. SYSOUT files that are not important are not deleted and will remain part of the output for the completed job.

System action:
This situation does not impact the function being performed by the workflow and has no effect on the final return code of the job. Processing continues.

User response:
If the situation occurs in an IBM-supplied workflow, then contact IBM Software Support.

KFU00115I **SYSOUT Application Program Interface (SAPI) failed; FUNCTION=SSS2(79) SSS2TYPE=*code* SSS2UFLG=*flag* RC=*return_code* SSOBRETN=*return_code* SSS2REAS=*reason_code* DDNAME=*name* CTOKEN=*string***

Explanation:
The JES subsystem interface SYSOUT application program interface (SAPI SSI=79) call was made to delete an output file that is not important. The request failed with the reported return and reason codes. The SYSOUT file is not deleted and will remain part of the output for the completed job.

System action:
This situation does not impact the function being performed by the workflow and has no effect on the final return code of the job. Processing continues.

User response:
If the situation occurs in an IBM-supplied workflow, then contact IBM Software Support.

KFU00116I **SYSOUT file delete request failed; DDNAME=*name* DSN=*name* STATUS=*code* CTOKEN=*string***

Explanation:
The output file is not important but could not be deleted. Status information is recorded for diagnostic purposes.

System action:
This situation does not impact the function being performed by the workflow and has no effect on the final return code of the job. Processing continues.

User response:
If the situation occurs in an IBM-supplied workflow, then contact IBM Software Support.

KFU00140E **REXX service has failed; Routine=IRXEXCOM RETC=*return code* VAR=*name***

Explanation:
The IRXEXCOM service was invoked to fetch or store REXX variable but failed with the reported return code. The service was invoked during KCIEXEC command processing of a **VGET** or **VPUT** request.

System action:
The REXX exec will throw an exception that either causes the exec to fail or give unpredictable results.

User response:
Contact IBM Software Support.

KFU00142E **Command syntax or parameter is not supported; REXX=*name* COMMAND=LISTDS**

Explanation:
The REXX exec issued the TSO/E **LISTDS** command. The request was intercepted and processed as a KCIEXEC command. A command parameter was not recognized.

System action:
The command is not processed, and the exec throws an exception. The results of the exec are unpredictable unless the exec handles error conditions.

User response:
If the REXX exec is supplied by IBM then contact IBM Software Support.

KFU00143E **Command syntax or parameter is not supported; REXX1=*name* REXX2=*name* VAR=*name* COMMAND=LISTDS**

Explanation:
This message is a variation of message KFU00142W. In this case, the REXX exec that issued the command is not the original EXEC REXX=*name* specified in the workflow. REXX1 is the original workflow exec and REXX2 is the most recently called exec.

System action:
The REXX exec will throw an exception that either causes the exec to fail or give unpredictable results.

User response:
Contact IBM Software Support.

KFU00144E **REXX service has failed; Routine=IRXEXCOM RETC=*return code* VAR=*name***

Explanation:

The IRXEXCOM service was invoked to set the value of a REXX variable but failed with the reported return code. The service was called by the **LISTDS** command.

System action:
The REXX exec will throw an exception that either causes the exec to fail or give unpredictable results.

User response:
Contact IBM Software Support.

KFU00145E DD statement is missing, DDNAME=*name*

Explanation:
The reported file, typically a system-generated ddname, is not allocated. The file was dynamically allocated but could not be found. This error should not occur in normal circumstances and indicates an environmental problem that KCIOMEGA cannot recover from. This error is associated with the processing of a TSO/E **CALL** command issued in a REXX exec.

System action:
The REXX exec will throw an exception that either causes the exec to fail or give unpredictable results.

User response:
Contact IBM Software Support.

KFU00147E Command syntax or parameter is not supported; REXX=*name* COMMAND=CALL

Explanation:
The REXX exec issued the TSO/E **CALL** command. The request was intercepted and processed as a KCIEXEC command. A command parameter was not recognized.

System action:
The command is not processed, and the exec throws an exception. The results of the exec are unpredictable unless the exec handles error conditions.

User response:
If the REXX exec is supplied by IBM then contact IBM Software Support.

KFU00148E Command syntax or parameter is not supported; REXX1=*name* REXX2=*name* COMMAND=CALL

Explanation:
This message is a variation of message KFU00147E. In this case, the REXX exec that issued the command is not the original EXEC REXX=*name* specified in the workflow. REXX1 is the original workflow exec and REXX2 is the most recently called exec.

System action:
The command is not processed, and the exec throws an exception. The results of the exec are unpredictable unless the exec handles error conditions.

User response:
If the REXX exec is supplied by IBM then contact IBM Software Support.

KFU00149E MVS service has failed; MACRO=LOAD PROGRAM=*name* R1=*system code* R15=*reason code*

Explanation:
The REXX exec issued the TSO/E **CALL** command. The request was intercepted and processed as a KCIEXEC command. The call program could not be loaded.

System action:
The command is not processed, and the exec throws an exception. The results of the exec are unpredictable unless the exec handles error conditions.

User response:
Verify that the program being called is in the call library. Otherwise if the REXX exec is supplied by IBM then contact IBM Software Support.

KFU00150E Command syntax or parameter is not supported; REXX=*name* PROBLEM=*reason* COMMAND=ALLOC

Explanation:
The REXX exec issued the TSO/E **ALLOC** command. The request was intercepted and processed as a KCIEXEC command. A command parameter was not recognized. Not all the actual TSO/E **ALLOCATE** options are supported. The reported problem describes the option that is not supported.

System action:
The command is not processed, and the exec throws an exception. The results of the exec are unpredictable unless the exec handles error conditions.

User response:
If the REXX exec is supplied by IBM then contact IBM Software Support.

KFU00151E Command syntax or parameter is not supported; REXX1=*name* REXX2=*name* PROBLEM=*reason* COMMAND=ALLOC

Explanation:
This message is a variation of message KFU00150E. In this case, the REXX exec that issued the command is not the original EXEC REXX= exec specified in the workflow. REXX1 is the original workflow exec and REXX2 is the most recently called exec.

System action:
The command is not processed, and the exec throws an exception. The results of the exec are unpredictable unless the exec handles error conditions.

User response:

If the REXX exec is supplied by IBM then contact IBM Software Support.

Index

Special Characters

A

B

C

D

E

G

I

J

K

M

O

P

R

S

T

U

V

Product Number: 5698-B66

SC28-3147