Version 1 Release 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: 2021-01-20

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 v

Tables vii

About this information ix

What's new in Monitoring Configuration Manager xi

Introduction 1
- Comparison with PARMGEN 2
- Prerequisites for using Configuration Manager 5
- Defining the OMEGAMON subsystem 6
- Creating your first, minimal RTE 7
- Creating or updating an RTE 14
  - Sharing runtime members or a full stand-alone set 17
  - Completing the parameters for discovered Db2 subsystems 17
  - Converting a hub to a remote monitoring server 18
  - Defining multiple RTEs in an RTEDEF library 20

Batch interface 25
- Actions 26
  - **CREATE** 27
  - **DISCOVER** 28
  - **GENERATE** 33
  - **DELETE** 34
  - **MIGRATE** 35
- Workflows 36
- GENERATE and maintenance scenarios 37

Parameters 39
- Initial parameters 39
  - **GBL_DSN_CICS_CTG_DLL** 39
  - **GBL_DSN_CSF_SCSFMOD0** 39
  - **GBL_DSN_DB2_DSNEXIT** 40
  - **GBL_DSN_DB2_LOADLIB_Vn** 40
  - **GBL_DSN_DB2_RUNLIB_Vn** 40
  - **GBL_DSN_IMS_RESLIB** 41
  - **GBL_DSN_IMS_SCEXLINK** 41
  - **GBL_DSN_IMS_SFUNLINK** 41
  - **GBL_DSN_WMQ_SCSQANLE** 42
  - **GBL_DSN_WMQ_SCSQAUTH** 42
  - **GBL_DSN_NETVIEW_CNMLINK** 42
  - **GBL_HFS_JAVA_DIRn** 42
  - **GBL_TARGET_HILEV** 43
  - **GBL_USS_TKANJAR_PATH** 43
  - **RTE_NAME** 44
  - **RTE_PLIB_HILEV** 44
  - **RTE_SECURITY_CLASS** 45
  - **RTE_SECURITY_FOLD_PASSWORD_FLAG** 45

RTE_SECURITY_USER_LOGON....46
RTE_STC_PREFIX....47
RTE_TCP_PORT_NUM....47
RTE_TEMS_NAME_NODEID....47
RTE_USS_RTEDIR....48
RTE_VTAM_APPLID_PREFIX....48
Parameters with significant default values....49
RTE_TYPE....49
Parameters introduced by Monitoring Configuration Manager....49
RTE_COMM_PROTOCOLn....49
RTE_TCP_KDEB_INTERFACELIST....51
RTE_TCP_PIPE6_PORT_NUM....52
RTE_TCP_PIPE6S_PORT_NUM....52
RTE_TCP_PIPES_PORT_NUM....52
RTE_TCP_STC....53
RTE_TCP_UDP_PORT_NUM....53
RTE_TCP_UDP6_PORT_NUM....54
RTE_VTAM_NETID....54
KFJ_SYSNAME....55
RTE_X_SECURITY_EXIT_LIB....55
RTE_X_OVERRIDE_EMBEDS_LIB....56
Parameters with different default values than PARMGEN....56
Sparse parameter tables: First row sets defaults for subsequent rows....57

RTEDEF library....61
Members....61
Member order....62
Initial members....63
rte_name....64
KDS$PARM....65
KGW$PARM....65
KQI$PARM....66
GBL$PARM....66

Runtime members....69

Communication between monitoring components....73

Variables....79

Using security exits....81

Using embed overrides....83

Troubleshooting....85

Messages....87
KFJ messages....87
KFU messages....91

Index....107

Figures

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

2. Actions 1

3. Overview of a minimal runtime environment 8

4. Example JCL to perform the CREATE action 8

5. CONFIGURE_* parameters for a minimal runtime environment 10

6. Parameters that specify z/OS-related identifiers for a minimal runtime environment 11

7. Example JCL to perform the DISCOVER action 12

8. Example JCL to perform the GENERATE action 12

9. Example JCL to perform the CREATE action 15

10. Before: A stand-alone runtime environment with a hub monitoring server 19

11. After: A runtime environment with a remote monitoring server 19

12. Defining two runtime environments in a single RTEDEF library 20

13. JCL to run Monitoring Configuration Manager 25

14. Example JCL to perform the CREATE action 27

15. Example JCL to perform the DISCOVER action 29

16. RTEDEF(KC5@lpar) member created by the DISCOVER action 31

17. RTEDEF(KD5@lpar) member created by the DISCOVER action 31

18. RTEDEF(KI5@lpar) member created by the DISCOVER action 32

19. RTEDEF(KMQ#lpar) member created by the DISCOVER action 32

20. RTEDEF(KN3@lpar) member created by the DISCOVER action 33

21. Example JCL to perform the GENERATE action 34

22. Example JCL to perform the DELETE action 35

23. Example JCL to perform the MIGRATE action 36

24. Initial RTEDEF(rte_name) member created by the CREATE action....64

25. Initial RTEDEF(KDS$PARM) member created by the CREATE action....65

26. Initial RTEDEF(KGW$PARM) member created by the CREATE action....65

27. Initial RTEDEF(KQI$PARM) member created by the CREATE action.... 66

28. Initial RTEDEF(GBL$PARM) member created by the CREATE action....66

29. How parameters affect the locations of runtime members....70

30. Typical topology of runtime environments in a sysplex....73

31. Parameters required to configure a typical topology....75

32. Parameters to configure communication between components, including significant default values....76

33. Example KCIPRINT output data set for a successful Monitoring Configuration Manager job....85

Tables

1. Monitoring Configuration Manager versus PARMGEN: overview 2

2. Monitoring Configuration Manager versus PARMGEN: details 4

3. Output of the DISCOVER action 28

4. Parameters created by the DISCOVER action 29

5. Members created by the DISCOVER action 30

6. Parameters with different default values in PARMGEN and Monitoring Configuration Manager 56

7. RTEDEF member naming convention 61

8. Characteristics of the initial runtime environment definition 63

9. RTEDEF members that define variables, and the LPARs to which they apply 80

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 25 for details.
- Select which parameters you want to use with Monitoring Configuration Manager. See“Parameters” on page 39 for a description of valid parameters.
- Specify a library to contain the runtime environment definitions (RTEDEF). See “Runtime environment definition library” on page 61 for more information.
- View troubleshooting material and error messages you might see when using Monitoring Configuration Manager. See “Troubleshooting” on page 85 and “Messages” on page 87 for more information.

What's new in Monitoring Configuration Manager

This history summarizes the changes in Monitoring Configuration Manager documentation.

Edition date	Description
December 2020	With APAR OA60562, the following products can now be configured using Monitoring Configuration Manager, regardless of them being part of a Suite or Pack offering or purchased as a standalone point product: • Tivoli Enterprise Monitoring Server 6.3.0 • OMEGAMON Enhanced 3270 User Interface 7.5.0 • IBM Z OMEGAMON Monitor for z/OS 5.6.0 • IBM OMEGAMON for z/OS 5.5.0 • IBM Z OMEGAMON Network Monitor 5.6.0 • IBM OMEGAMON for Networks on z/OS 5.5.0 • IBM Z OMEGAMON Integration Monitor 5.6.0 • IBM OMEGAMON for Db2 Performance Expert 5.4.0 • IBM OMEGAMON for Db2 Performance Monitor on z/OS 5.4.0 • IBM OMEGAMON for CICS on z/OS 5.5.0 • IBM OMEGAMON for IMS on z/OS 5.5.0 • IBM OMEGAMON for Messaging on z/OS 7.5.0 • IBM Z OMEGAMON for JVM 5.5.0 • IBM Z OMEGAMON Runtime Edition for JVM 5.5.0 • IBM OMEGAMON for Storage on z/OS 5.5.0 • IBM OMEGAMON for Messaging on z/OS 7.5.0 • IBM Z NetView Agent 6.3.0 • ITCAM for Application Diagnostics Agent 7.1.0 • IBM OMEGAMON Dashboard Edition on z/OS 5.5.0
November 2020	Monitoring Configuration Manager updates include the following new features: • MIGRATE action as described in “MIGRATE” on page 35. • Embed overrides as described in “Using embed overrides” on page 83. • Security exits as described in “Using security exits” on page 81.
July 2020	First public edition, coinciding with the general availability (GA) of the fix for APAR OA59463, PTF UJ03556 (new function: Monitoring Configuration Manager).

Introduction to Monitoring 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 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:

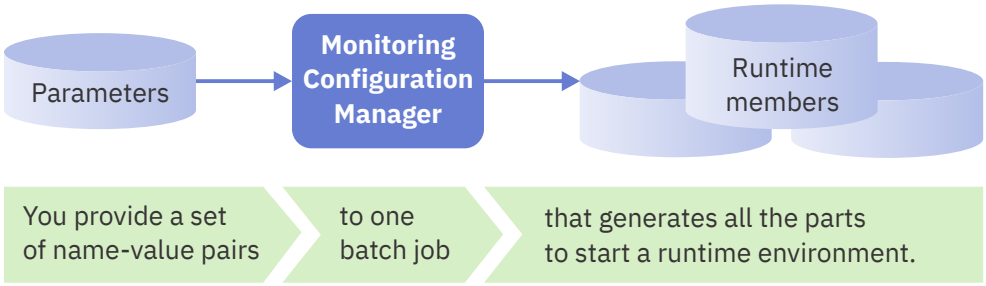

Figure 1. The basic concept: parameters in, one job, runtime members out

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.

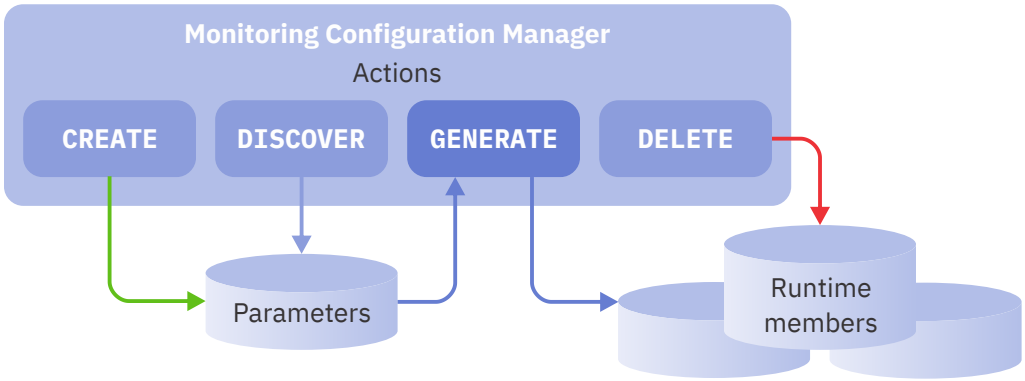

Figure 2. Actions

In addition to the actions shown in the diagram above, the MIGRATE action can be used as needed to import existing legacy PARMGEN RTE configuration settings from a specific WCONFIG member into the new "batch only" Monitoring Configuration Manager dataset. See “MIGRATE” on page 35 for more information.

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 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.

Monitoring Configuration Manager versus 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 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.

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.

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

Monitoring Configuration Manager	PARMGEN
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.
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 the z/OS Discovery Library Adapter (DLA).
Only available for all agents of the IBM Z Monitoring Suite and the IBM Z Netview Monitoring agent, as well as for IBM OMEGAMON for z/OS version 5.5.0 and higher, IBM OMEGAMON for Networks on z/OS V5.5.0 and higher, and IBM Z OMEGAMON Integration Monitor V5.5.0 and higher. 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 56.

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

Requires the TSO user running a Monitoring Configuration Manager job to have the same permissions as for PARGMEN jobs, except: **no z/OS superuser authority required**.	Requires the TSO user running PARMGEN jobs to have the following permissions:
	• Read access to the following z/OS System Authorization Facility (SAF) resources in the FACILITY class:
	`BPX.FILEATTR.APF`
	`BPX.FILEATTR.PROGCTL`
	• z/OS superuser authority (UID=0). For example, read access to the `BPX.SUPERUSER` resource.

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.	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 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.

To assist with moving legacy PARMGEN data to Configuration Manager, a new MIGRATE action has been created. This command allows you to import existing legacy PARMGEN RTE configuration settings into a "batch only" configuration dataset. For more information, see “MIGRATE” on page 35.

Prerequisites for using Configuration Manager

To use IBM Z Monitoring Configuration Manager, you need to know where your SMP/E installation target library is located. In addition, the TSO userid that will be used to run Monitoring Configuration Manager jobs needs some special access privileges.

Location of 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: 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

The TSO user ID that you plan to run Monitoring 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 Monitoring Configuration Manager.

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:

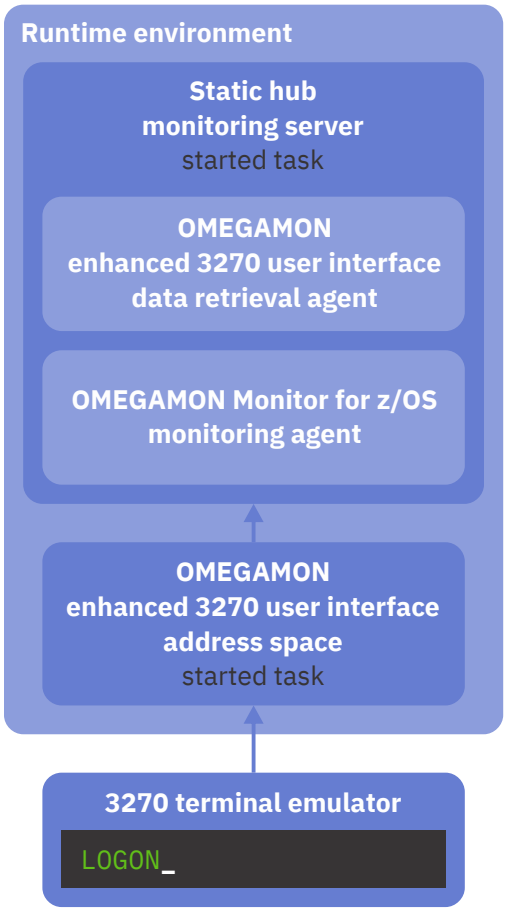

Figure 3. Overview of a minimal runtime environment

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 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 44-character limit for data set names, the combined length of **RTE_NAME** and **RTE_PLIB_HILEV** should not exceed 18 characters. For example, if **RTE_NAME** is 8 characters, then **RTE_PLIB_HILEV** should not exceed 10 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:

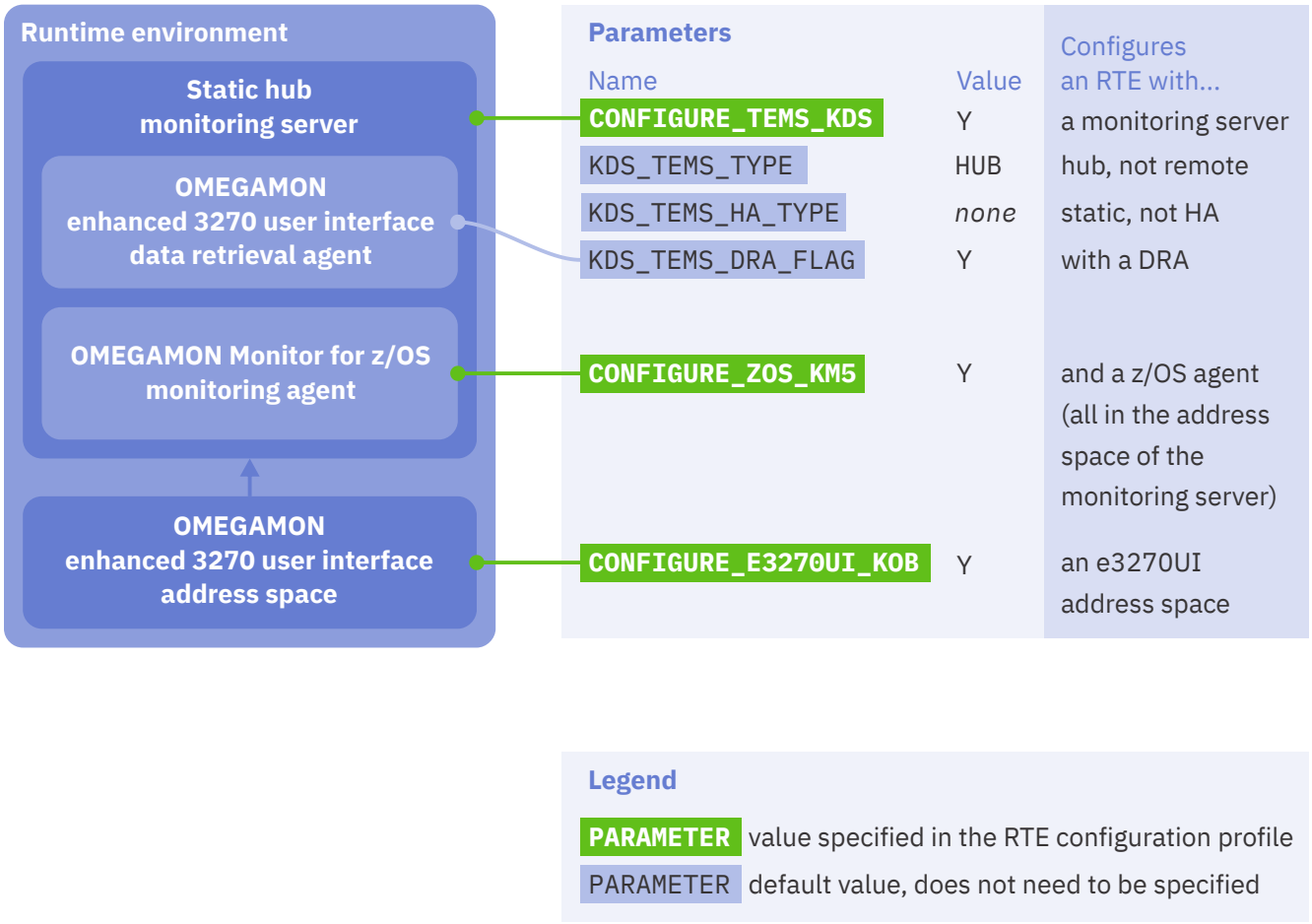

*Figure 5. **CONFIGURE_*** parameters for a minimal runtime environment*

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 OM*xx*OBAP. For example, if the system (LPAR) name is Z0S1, then the VTAM applid is OMS10BAP.

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.

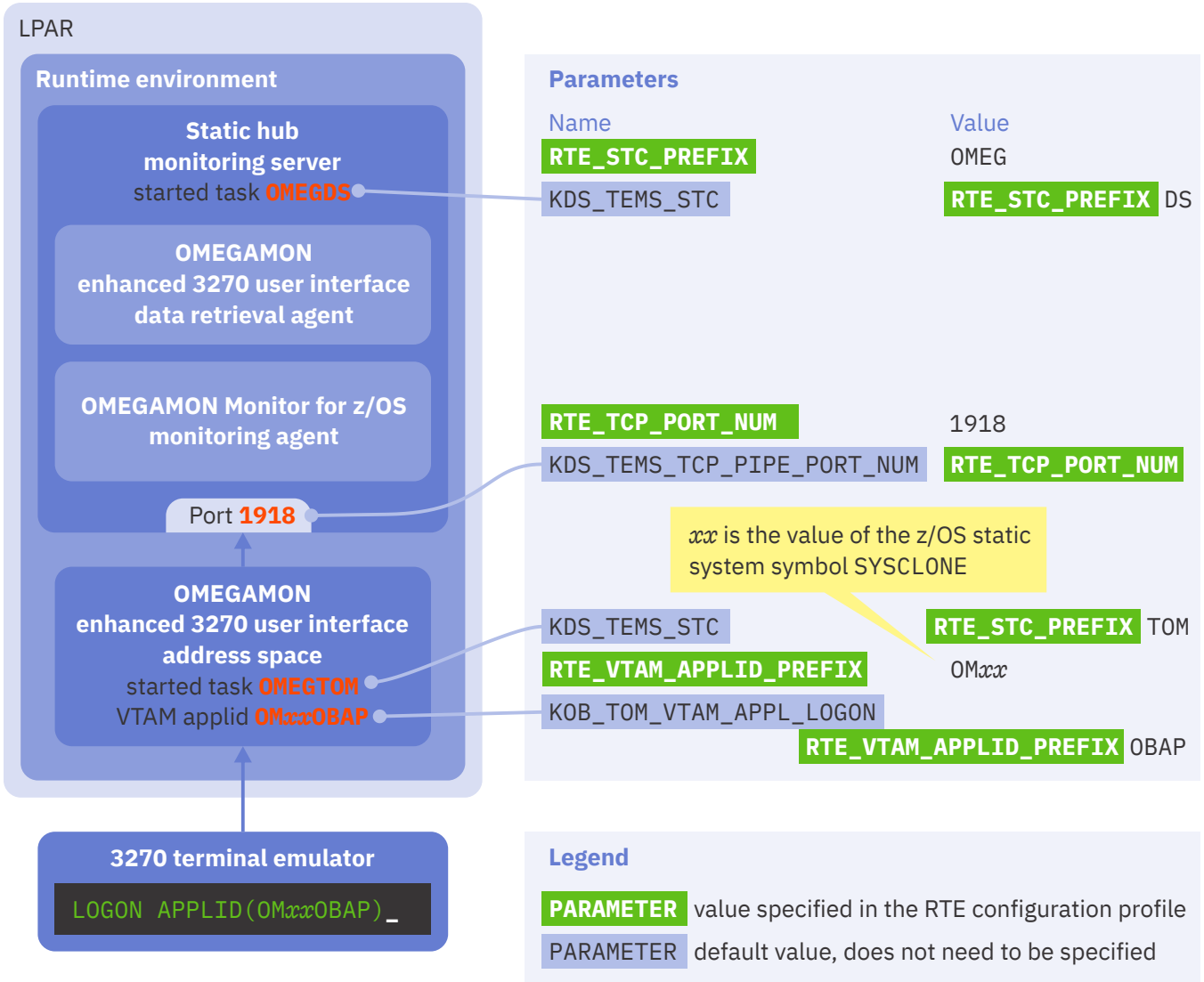

Figure 6. Parameters that specify z/OS-related identifiers for a minimal runtime environment

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

The **DISCOVER** action discovers CICS regions, Db2 subsystems, IMS control regions, MQ subsystems, and TCP/IP stacks, 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
To troubleshoot issues with Monitoring Configuration Manager jobs, use a tool such as SDSF to view the JES output data sets.

Prerequisites for using Configuration Manager
To use IBM Z Monitoring Configuration Manager, you need to know where your SMP/E installation target library is located. In addition, the TSO userid that will be used to run Monitoring Configuration Manager jobs needs some special access privileges.

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 library" on page 61.

   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 25.

   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 and TCP/IP stacks, 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 17.

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.

   > **Tip:** Where instructions refer to using PARMGEN, use Monitoring Configuration Manager:
   >
   > - Instead of using the PARMGEN Workflow user interface to set parameters, edit the corresponding RTEDEF library members.
   > - Instead of submitting one or more PARMGEN jobs, submit a job that performs the **GENERATE** action.

7. For a newly created runtime environment: start the tasks. For an updated runtime environment: stop and then restart the tasks.

Related concepts

Troubleshooting

To troubleshoot issues with Monitoring Configuration Manager jobs, use a tool such as SDSF to view the JES output data sets.

Prerequisites for using Configuration Manager
To use IBM Z Monitoring Configuration Manager, you need to know where your SMP/E installation target library is located. In addition, the TSO userid that will be used to run Monitoring Configuration Manager jobs needs some special access privileges.

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.

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

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.

The **DISCOVER** action sets the **KD2_DBnn_DB2_USEFLG** parameter to N. This value causes the **GENERATE** action to ignore the **KD2_DBnn_DB2_LOADLIB** and **KD2_DBnn_DB2_RUNLIB** parameters, and instead to use the value of the **KD2_DBnn_DB2_VER** parameter to get the appropriate version-specific values from the global parameters **GBL_DSN_DB2_LOADLIB_Vn** and **GBL_DSN_DB2_RUNLIB_Vn**.

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, set **KD2_DBnn_DB2_USEFLG** to Y and specify a value for **KD2_DBnn_DB2_RUNLIB**.

Related reference

DISCOVER

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

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:

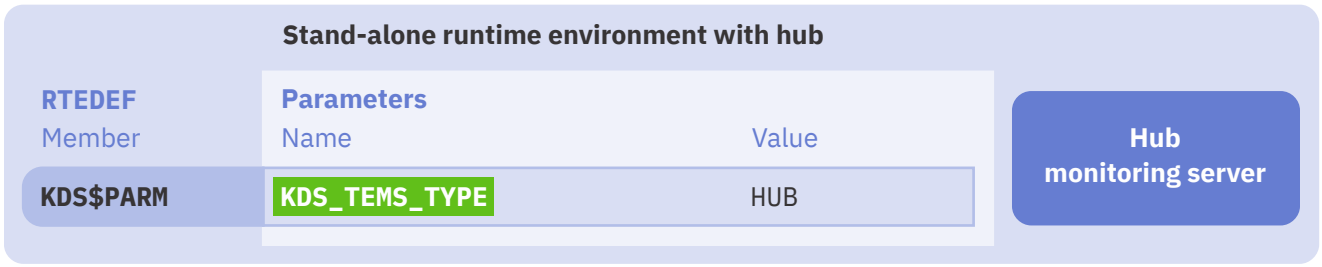

Figure 10. Before: A stand-alone runtime environment with a hub monitoring server

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

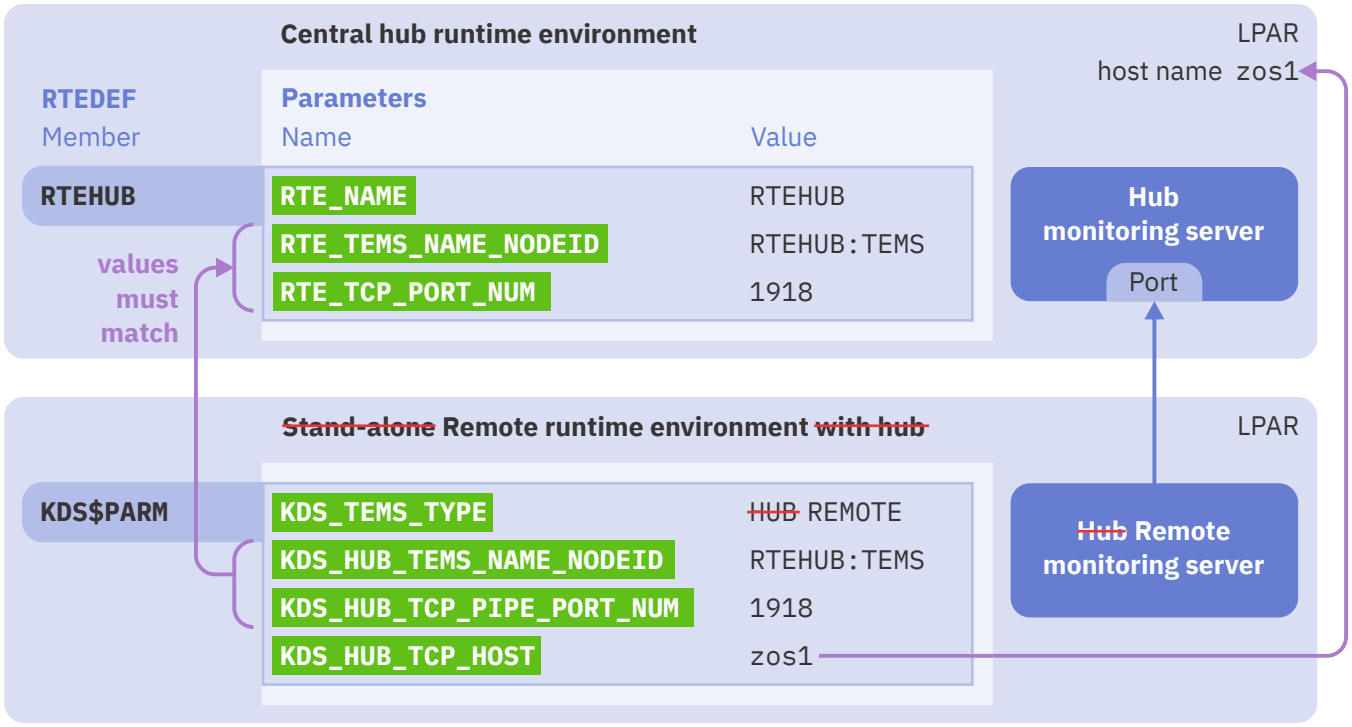

Figure 11. After: A runtime environment with a remote monitoring server

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:

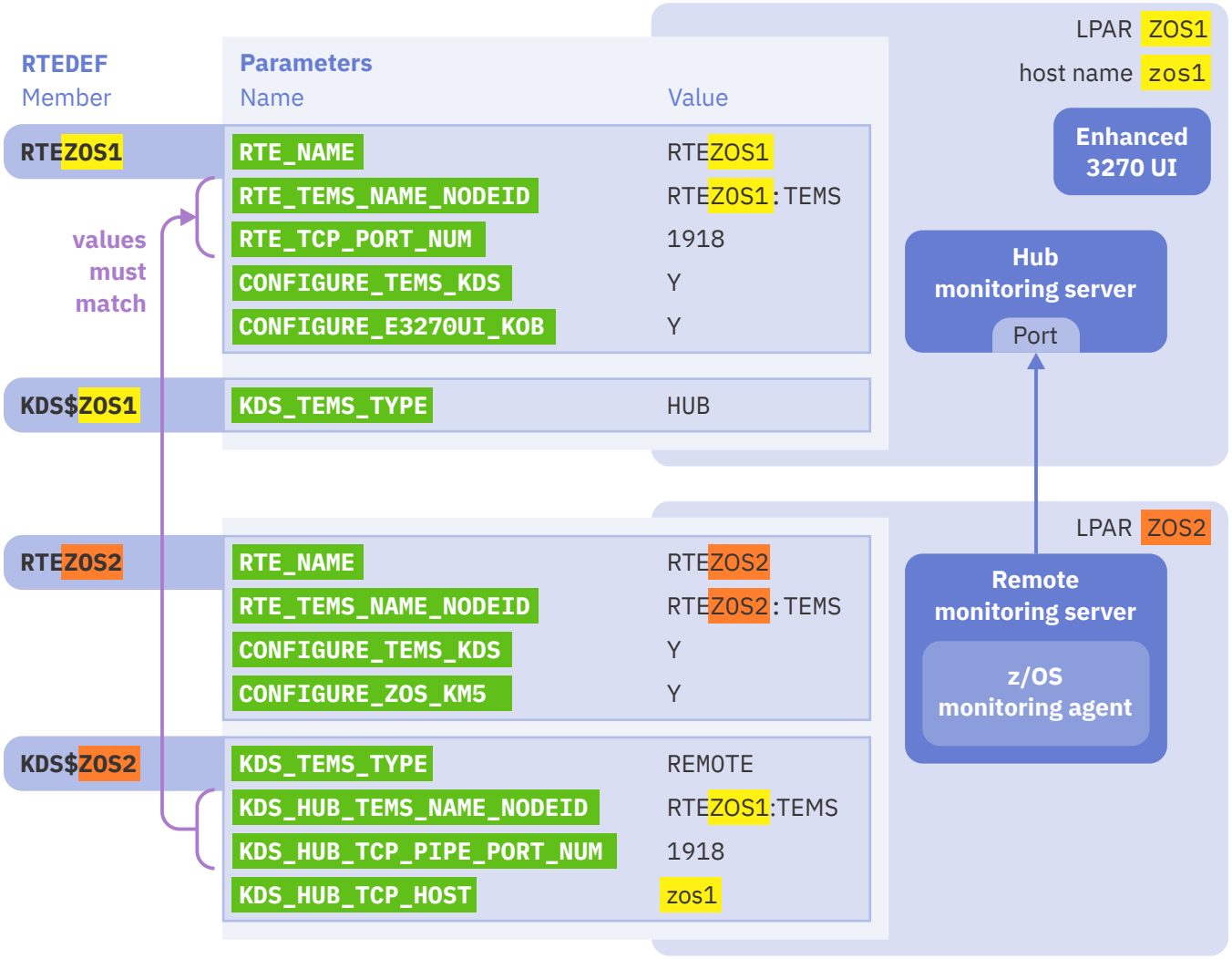

Figure 12. Defining two runtime environments in a single RTEDEF library

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, RTEZOS2.

5. Edit the new RTEDEF(RTEZOS2) 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(RTEZOS1) 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, RTEZOS2.

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            <action> * CREATE|DISCOVER|GENERATE|DELETE|MIGRATE
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 36.

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**, and **MIG**.

<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 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 44-character limit for data set names, the combined length of **RTE_NAME** and **RTE_PLIB_HILEV** should not exceed 18 characters. For example, if **RTE_NAME** is 8 characters, then **RTE_PLIB_HILEV** should not exceed 10 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 the legacy PARMGEN for use with Monitoring Configuration Manager. When using MIGRATE, the order of actions would be the following:

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

CREATE

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

The **CREATE** action allocates the runtime environment definition library, *rte_plib_hilev*.RTEDEF, if it does not already exist, and the Security exits library *<rte_plib_hilev>.<rte_name>*.SECEXITS. Then **CREATE** populates the RTEDEF and SECEXITS libraries with an initial set of configuration profile members.

CREATE does not overwrite members

If the RTEDEF library already exists, **CREATE** only writes members that do not yet exist.

CREATE is optional: you can do it yourself

CREATE is an optional convenience for new users. Experienced users can skip **CREATE** and copy an existing RTEDEF library.

You can allocate the RTEDEF library yourself:

Record format
Fixed-length, blocked (FB)

Record length
80

You can also create the members. The only required member is *rte_name*.

Example

The following JCL creates the runtime environment definition library TSOUID.MONSUITE.RTEDEF and populates it with various members, including the runtime environment configuration profile member 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            CREATE
RTE_NAME          RTE1
RTE_PLIB_HILEV    TSOUID.MONSUITE
/*
```

*Figure 14. Example JCL to perform the **CREATE** action*

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.

Using security exits

Several security exits are used in the legacy PARMGEN and can also be customized for use by Monitoring Configuration Manager. You can use the MIGRATE action to import the legacy PARMGEN Security Exits into a new data set as part of the RTEDEF dataset, and modify them as needed.

DISCOVER

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

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.

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.

Review the messages in the KCIPRINT sysout data set

After performing a **DISCOVER** action, review the messages about discovery in the KCIPRINT sysout data set.

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 that you can use in the KMQUSER monitoring file: `SET MANAGER NAME(queue-manager)`

Example

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=KCIOMEGA,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 15. 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.

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.

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**	Yes	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*
TCP/IP	KN3@*lpar* for first-time discovery KN3#*lpar* for rediscovery

Member naming convention

The **DISCOVER** action creates RTEDEF library members with the following naming convention:

Prefix
Identifies the type of subsystem: K*pp* from the corresponding **CONFIGURE_*** parameter.

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

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

Suffix
Identifies the LPAR.

First-time discovery versus rediscovery

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

If a K*pp*@*lpar* member already exists, the **DISCOVER** action writes a comment member, K*pp*#*lpar*, 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 desired 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

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

KC2_HS                          END                * Table end *
```

*Figure 16. RTEDEF(KC5@lpar) member created by the **DISCOVER** action*

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"
KD2_DB01_DB2_DSNTIAD       "DSNTIAD"
KD2_DB01_DB2_USEFLG        "N"
KD2_DB01_DB2_RUNLIB        ""

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

KD2_DB                     END                       * Table end *
```

*Figure 17. 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.

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"
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 18. 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.

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 KMQUSER monitoring file.

```
* 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 19. RTEDEF(KMQ#lpar) member created by the **DISCOVER** action*

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"
KN3_TCPX01_TCPIP_PROFILES_MBR     "member-name"

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

KN3_TCPX                          END                * Table end *
```

*Figure 20. RTEDEF(KN3@lpar) member created by the **DISCOVER** action*

GENERATE

The **GENERATE** action generates runtime members from the parameters in the runtime environment definition library, *rte_plib_hilev*.RTEDEF.

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 (PDS) data sets. Locked runtime members can cause the **GENERATE** action to fail.

The **GENERATE** action builds the set of parameters for a runtime environment 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, if the RTEDEF library contains the following members:

```
KDS$PARM
KDS$ZOS1
KDS$ZOS2
```

and 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.

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 21. Example JCL to perform the **GENERATE** action*

Note the JES2 **SYSAFF** 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 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.

DELETE

The **DELETE** action deletes the runtime libraries for the runtime environment.

The **DELETE** action also deletes the "legacy" PARMGEN libraries that the **GENERATE** action creates: WCONFIG, interim staging, work, and global user JCL.

Required CONFIRM parameter

When you specify the **DELETE** action in the KCIVARS data set, you must also specify a **CONFIRM** workflow variable with the value Y:

```
CONFIRM Y
```

Name patterns of data sets deleted

The **DELETE** action deletes 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*.

Locations *not* affected by DELETE

The **DELETE** action does not affect the following locations:

- The runtime environment definition library, *rte_plib_hilev*.RTEDEF
- The Security Exits library , <*rte_plib_hilev*>.<*rte_name*>.SECEXITS
- The Embed Overrides library <*rte_plib_hilev*>.<*rte_name*>.EMBEDS
- The "staging" system VTAMLIB, VTAMLST, and PROCLIB libraries: *rte_hilev*.SYS1.*

- Any other data sets, such as persistent data store (PDS) data sets, that have been allocated outside the *rte_name*-based data set name patterns in the previous list
- z/OS UNIX directories

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 22. 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.

MIGRATE

The **MIGRATE** action enables you to import existing legacy PARMGEN RTE configuration settings from a specific WCONFIG member into the new "batch only" Monitoring Configuration Manager *rte_plib_hilev*.RTEDEF.

MIGRATE process

Before performing a **MIGRATE** action, make sure you have a backup of your existing legacy PARMGEN RTE in place.

In general, to migrate a legacy PARMGEN runtime environment (RTE) to a Monitoring Configuration Manager RTE, follow these steps:

1. Modify the KFJJMCM sample job in TKANSAM (see example below) to select a MIGRATE action.
2. Specify the source WCONFIG in the KFJ_MIGRATE_WCONFIG parameter.
3. Run the KFJJMCM job to perform the migration and generate the new RTEDEF dataset.

   The MIGRATE action reads WCONFIG and other datasets from a former legacy PARMGEN installation and creates the sparse descriptors for it 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.

Note: If you want to migrate embed overrides, see “Using embed overrides” on page 83 for more information.

The MIGRATE action accepts PARMGEN RTEs with system variables. However, system variables are *not* copied unless you have chosen to override them in your legacy PARMGEN configuration. Variables are copied to the RTEDEF member VAR$GLOB.

If MIGRATE detects that a specified target RTEDEF dataset already exists, MIGRATE issues an error message and stops. If you want to use the existing RTEDEF dataset, change the CONFIRM parameter in

KCIVARS to "Y". In this case, MIGRATE will first delete the existing RTEDEF dataset and then re-create it using the migrated parameters.

The MIGRATE action supports migrating one RTE at the time. It does not support migrating multiple legacy PARMGEN RTEs into a single RTEDEF configuration. Therefore the MIGRATE action will create members of type "Kpp$PARM" in the respective created RTEDEF data set, along with the *rte_name* member for the RTE-specific parameters.

The migration will work only for the OMEGAMON products that are supported by Monitoring Configuration Manager. If the migration source contains other products configured by the legacy PARMGEN that are not supported by Monitoring Configuration Manager, a warning message is issued as part of the MIGRATE action in the KCIPRINT output, but the MIGRATE action will continue to perform as expected for supported products.

Note: After MIGRATE, the generated RTEDEF dataset members must be carefully checked to see if the parameters have the expected values. Extra care must be taken with dataset high-level qualifiers, dataset names, and USS paths, as a subsequent GENERATE action might overwrite existing files.

Example of MIGRATE JCL

The following JCL is used to MIGRATE an existing legacy PARMGEN configuration pointed to by *highlevel*.WCONFIG into an RTEDEF library TSOUID.MONSUITE.RTEDEF.

During the MIGRATE action, the contents of the RTEDEF dataset will contain the respective parameters for the products configured in the legacy PARMGEN source.

```
//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 * CREATE|DISCOVER|GENERATE|DELETE|MIGRATE
RTE_NAME           RTE1
RTE_PLIB_HILEV     TSOUID.MONSUITE

* Required for the DELETE/MIGRATE actions, ignored by other actions:
CONFIRM               N  * Y|N
KFJ_MIGRATE_WCONFIG   'highlevel.WCONFIG'

/*
```

*Figure 23. Example JCL to perform the **MIGRATE** action*

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*.

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 33.

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 in the OMEGAMON shared documentation.

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

The Integrated Cryptographic Service Facility (ICSF) load library that contains the CSNB* modules used for password encryption.

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 RESLIB library.

The IMS RESLIB 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.

> **Tip:** 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 RESLIB library. Depending on how IMS is configured at your site, the same value might be appropriate for **GBL_DSN_IMS_RESLIB**.

Required

Required if the runtime environment configures the IMS monitoring agent.

Default value

DFS.V12R0M0.SDFSRESL

Values

An MVS data set name.

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 Netview CNMLINK library

Required

It is required only if using a NetView agent.

Default value

NETVIEW.VNRNMN.CNMLINK

Values

An MVS data set name.

GBL_HFS_JAVA_DIRn

The z/OS UNIX 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 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.

This is a read-only directory for Monitoring Configuration Manager; Monitoring Configuration Manager does not write to this directory.

Required

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

none

Values

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

Example

```
/usr/lpp/kan/bin/IBM
```

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 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 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.

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 44-character limit for data set names, the combined length of **RTE_NAME** and **RTE_PLIB_HILEV** should not exceed 18 characters. For example, if **RTE_NAME** is 8 characters, then **RTE_PLIB_HILEV** should not exceed 10 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 paths) that are specified by parameters.

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

RTE_SECURITY_CLASS

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

The **RTE_SECURITY_CLASS** parameter applies only to the OMEGAMON enhanced 3270 user interface. To secure other products, see the product-specific documentation.

Required?

No

Default value

none

Runtime members

See the **RTE_SECURITY_CLASS** parameter in the KPPENV member of the RKANPARU library.

Values

A SAF class name: a string of up to 8 characters. If you are using ACF2 as your external security resource manager, specify a maximum of 3 characters.

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_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.

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 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.

Here are some basic parameters that are not included in the initial set, but whose default values significantly affect 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

IBM Z Monitoring Configuration Manager: SHARING

PARMGEN: FULL

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.

Related tasks
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.

Parameters introduced by Monitoring Configuration Manager

IBM Z Monitoring Configuration Manager introduces some parameters that do not exist in PARMGEN. 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. In addition, the **KFJ_SYSNAME** parameter is also described below.

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.

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.

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 CREATE, DISCOVER, and GENERATE 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.

Required?

This is only required if you must override the default value.

RTE_X_SECURITY_EXIT_LIB

This parameter specifies the library containing the security exits used for the runtime environment.

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 (KOBSUPDT OMEGAMON KppSUPDI exits; Tivoli Monitoring Services: Engine security exits, external security exits, etc.).

Required?

No

Default value

<rte_plib_hilev>.<rte_name>.SECEXITS

Values

A valid MVS data set name.

Example

TEST1.TST.DEMO.SECEXITS

RTE_X_OVERRIDE_EMBEDS_LIB

The RTE_X_OVERRIDE_EMBEDS_LIB parameter specifies the name of the source library for embed members.

Embed override files can be used to add user-defined parameters that might be overwritten during the **GENERATE** action. This parameter is automatically defined in *<rte_plib_hilev>*.RTEDEF(*rte_name*) member during **CREATE** or **MIGRATE** actions, if the KFJ_USE_EMBEDS parameter is set to "Y".

This library contains all the 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

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 6. 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_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

Table 6. 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
KMQ_STARTMON_ACTIVEONLY	NO Only monitoring active queue managers is best practice.	YES
KYN_XAI01_SUBAGENT_ JAVAHOME	/usr/lpp/java/J7.1 There's 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
RTE_USS_RTEDIR	/rtehome Creating a new subdirectory of the root directory is bad practice.	/var/rtehome
RTE_X_SECURITY_EXIT_LIB	*ret_hilev.retname*.RKANSAMU Identifies the security exits library currently used in the legacy PARMGEN.	*rte_hilev.rte_name* .SECEXITS

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_DB01_DB2_USEFLG        "N"

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_DB03_DB2_USEFLG         "Y"

KD2_DB                      END
```

The first row configures the agent to monitor the DB2P subsystem and sets the default values for subsequent rows (subsystems).

The **KD2_DB01_DB2_USEFLG** parameter value N causes the **GENERATE** action to ignore the **KD2_DB01_DB2_LOADLIB** and **KD2_DB01_DB2_RUNLIB** parameters, which have been deliberately omitted from this example. Instead, the **GENERATE** action uses the value of the **KD2_DB01_DB2_VER** parameter to get the appropriate version-specific values from the global parameters **GBL_DSN_DB2_LOADLIB_Vn** and **GBL_DSN_DB2_RUNLIB_Vn**.

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 sets the **KD2_DB03_DB2_USEFLG** parameter to Y, and specifies its own values for LOADLIB and RUNLIB.

Runtime environment definition 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.

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 7. 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.
K*pp*$PARM	**Kpp_***	All	Product configuration profile.
K*pp*$*lpar*	**Kpp_***	Current	LPAR-specific product configuration profile.
GBL$PARM	**GBL_***	All	Global configuration profile.
GBL$*lpar*	**GBL_***	Current	LPAR-specific global configuration profile.

Supported values of *pp* in K*pp*$PARM and K*pp*$*lpar*

The **GENERATE** action processes K*pp*$PARM and K*pp*$*lpar* member names that contain the following values of *pp*:

C5, D5, DS, GW, I5, JJ, M5, MQ, NA, N3, OB, QI, S3, WO, YN

For details of the parameters that you can specify in each member, see https://www.ibm.com/support/knowledgecenter/SSAUBV/com.ibm.omegamon_share.doc_6.3.0.2/zcommonconfig/configparms_wheretofind.htm in the OMEGAMON shared documentation.

For product-specific member name prefixes, use the suffix of the CONFIGURE_* parameter

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.

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.

1. *rte_name*
2. K*pp*@*lpar*
3. K*pp*$PARM
4. K*pp*$*lpar*
5. GBL$PARM
6. GBL$*lpar*

(RTE, product, global)

where *lpar* identifies the LPAR on which the **GENERATE** action runs.

If a parameter is set more than once, the *last* value is used.

LPAR-specific members take precedence:

- Parameters in GBL$*lpar* override GBL$PARM
- Parameters in K*pp*$*lpar* override K*pp*$PARM

Parameters in K*pp*$* members override the K*pp*@* members created by the **DISCOVER** action.

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 8. 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 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)
- MQ Integration Broker (KQI$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.

Related reference

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

Members created by the DISCOVER action
The **DISCOVER** action creates members for each type of subsystem it discovers.

Runtime environment definition 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_CLASS
<rte_plib_hilev>.<rte_name>.SECEXITS
RTE_TEMS_NAME_NODEID              <rte_name>:TEMS
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 Agent
CONFIGURE_MFN_KN3                 Y  * Network
CONFIGURE_STORAGE_KS3             Y  * Storage
CONFIGURE_OMEGAVIEW_KWO           Y  * Integration Monitor
CONFIGURE_ITCAMAD_KYN             Y  * ITCAM for Applications
```

*Figure 24. Initial RTEDEF(rte_name) member created by the **CREATE** action*

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 25. 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 26. Initial RTEDEF(KGW$PARM) member created by the **CREATE** action*

RTEDEF(KQI$PARM)

The RTEDEF(KQI$PARM) member contains parameters that configure the MQ Integration Broker monitoring agent. These parameters have the prefix **KQI**.

```
* Integration Broker

KQI_XML_XIMBDIR1                  "/var/wmqi/YOURBKR"
KQI_XML_XIMBNAME_MON_BRKR_NAME    "YOURBKR"
```

*Figure 27. Initial RTEDEF(KQI$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 28. 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.

Runtime members

The **GENERATE** action generates runtime members in locations (MVS data set names and z/OS UNIX 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:

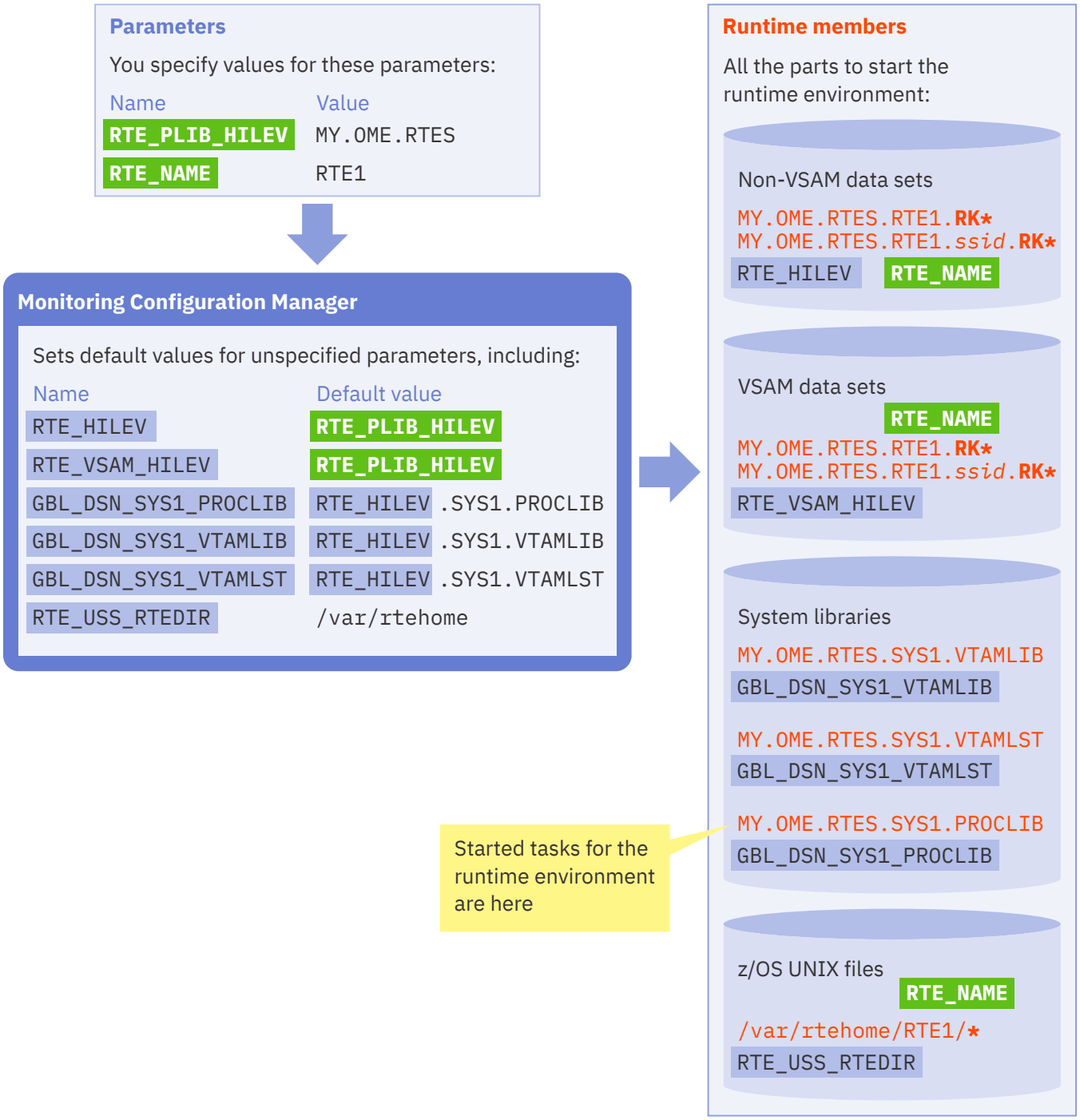

Figure 29. How parameters affect the locations of runtime members

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 reference
GENERATE

The **GENERATE** action generates runtime members from the parameters in the runtime environment definition library, *rte_plib_hilev*.RTEDEF.

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.

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.

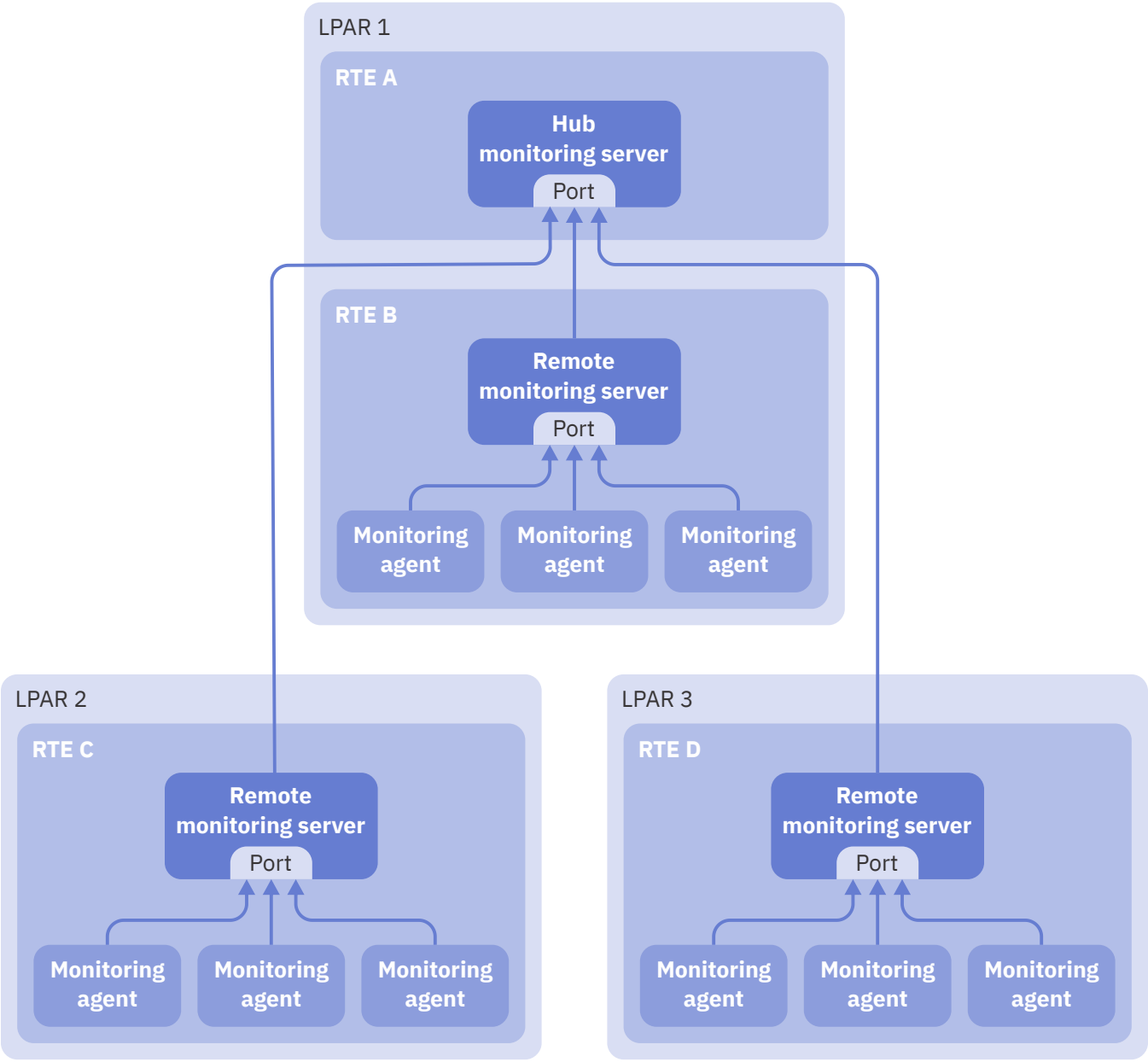

Figure 30. Typical topology of runtime environments in a sysplex

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:

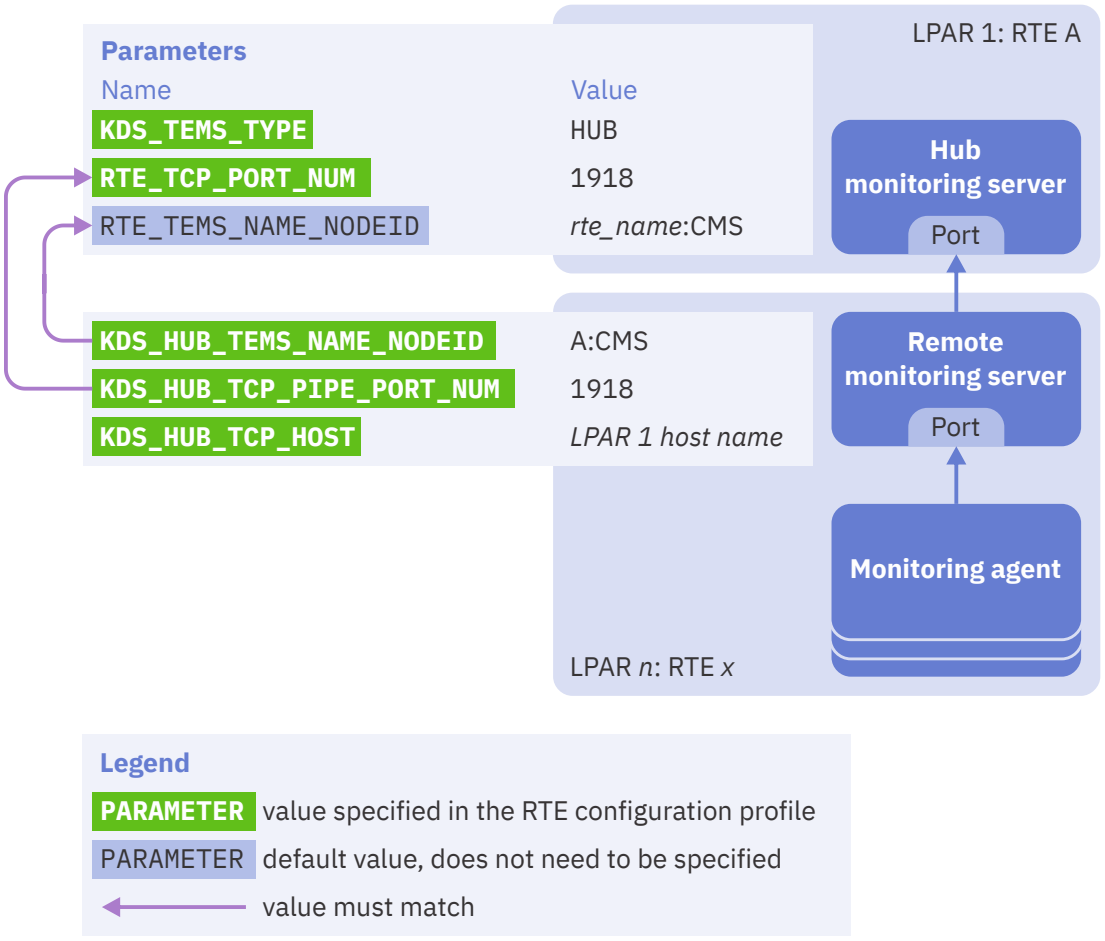

Figure 31. Parameters required to configure a typical topology

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.

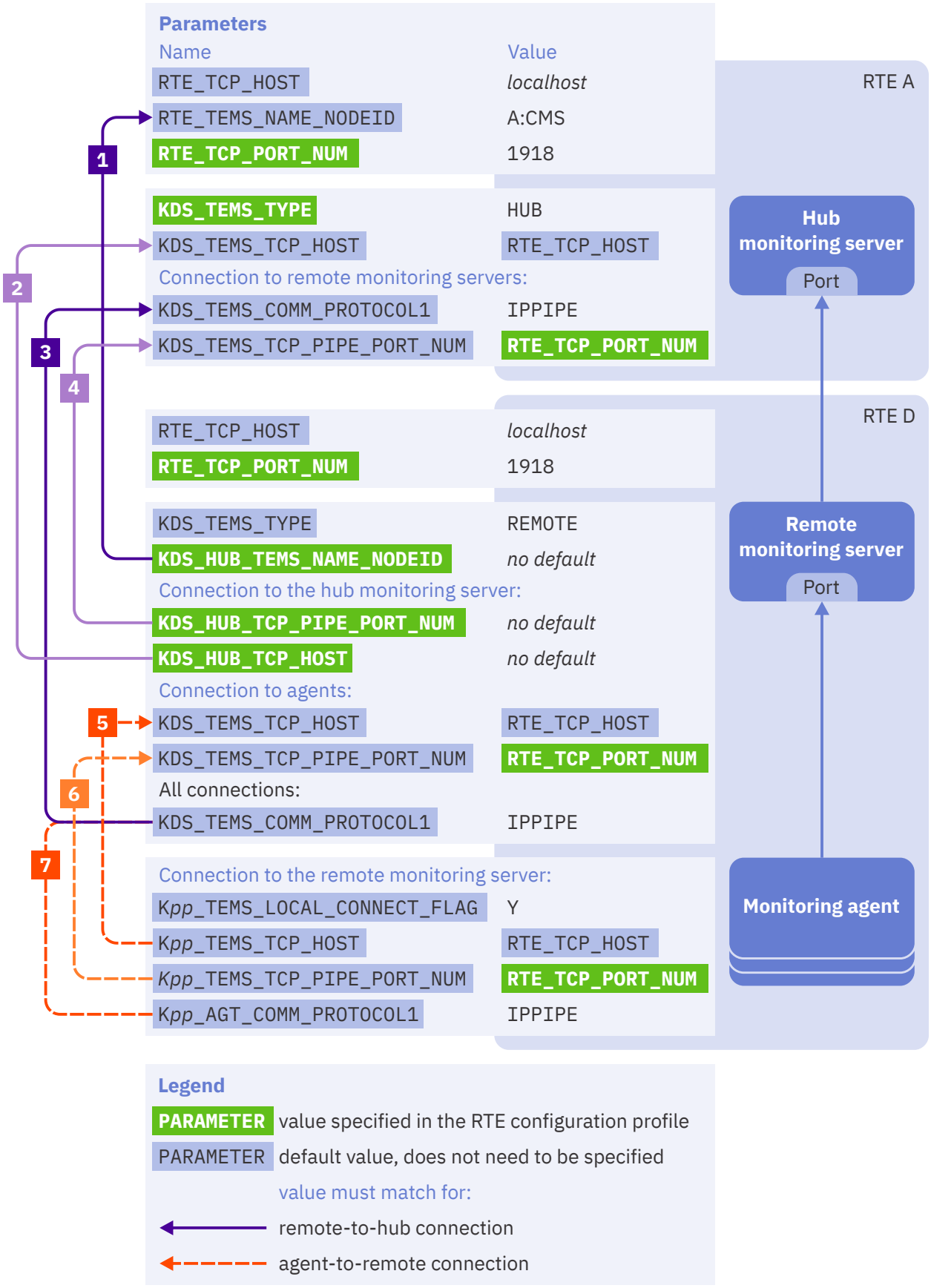

Figure 32. Parameters to configure communication between components, including significant default values

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 9. 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.

Using security exits

Several security exits are used in the legacy PARMGEN and can also be customized for use by Monitoring Configuration Manager. You can use the MIGRATE action to import the legacy PARMGEN Security Exits into a new data set as part of the RTEDEF dataset, and modify them as needed.

Required security exits

The security exits currently in use for the legacy PARMGEN will also be used by Monitoring Configuration Manager. The exits can be found in the SECEXITS dataset (example below) and can be modified as needed.

The use of security exits is required, however, you can modify the exits as needed. The default path is *<rte_plib_hilev>.<rte_name>*.SECEXITS, where:

- *rte_plib_hilev* is the high-level qualifier for the library
- *rte_name* is the member name

Use ISPF to display a list of members in the *<rte_plib_hilev>.<rte_name>*.SECEXITS library. Each of these members can be modified for your environment, if needed.

Security exits and the CREATE job

When the CREATE job is run, it automatically creates two libraries for use by Monitoring Configuration Manager:

- SECEXITS – Contains the required security exits library. The default library is *<rte_plib_hilev>.<rte_name>*.SECEXITS.
- RTEDEF – Creates a required reference to the security exits library. The default library is *<rte_plib_hilev>*.RTEDEF(*rte_name*).

The use of SECEXITS is mandatory, but you have the option to modify one or more of the exits as needed.

Use the KFJ_SECURITY_EXITS_LIB parameter in the CREATE job to specify the library for the security exits. You can either use the default library, which is *<rte_plib_hilev>.<rte_name>*.SECEXITS, for security exits or you can specify a different dataset name in the CREATE job:

```
KFJ_SECURITY_EXITS_LIB dataset_name
```

An example of this new command in the CREATE job is shown below.

```
//KCIVARS DD *
ACTION           CREATE * CREATE|DISCOVER|GENERATE|DELETE|MIGRATE
RTE_NAME         DEMO
RTE_PLIB_HILEV   TEST1.TST

KFJ_SECURITY_EXITS_LIB  'TEST1.TST.DEMO.EXITS'
```

In this example, the KFJ_SECURITY_EXITS_LIB parameter is set to TEST1.TST.DEMO.EXITS.

Note: If this dataset does not exist, it will be created. If the dataset already exists, security exit members will be copied to the existing library (but not overwritten).

In addition, *<rte_plib_hilev>*.RTEDEF(*<rte_name>*) will contain a reference (parameter)

```
RTE_X_SECURTY_EXITS_LIB TEST1.TST.DEMO.EXITS
```

that points to the security exit library.

The KFJ_SECURITY_EXITS_LIB parameter works with the CREATE and MIGRATE actions only.

Security Exits and the MIGRATE job

The MIGRATE job is similar to the CREATE job, in that the result will include the <*rte_plib_hilev*>.<*rte_name*>.SECEXITS library and the RTEDEF library. When the MIGRATE job is run, Monitoring Configuration Manager copies the required security exits used by the legacy PARMGEN to the newly created SECEXITS library, which is <*rte_plib_hilev*>.<*rte_name*>.SECEXITS.

Note: The legacy PARMGEN dataset containing the security exits is not changed as a result of the MIGRATE job.

An example of how to specify the *security exit library* for MIGRATE is shown below.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//* ****************************************************************
//* IBM Z Monitoring Configuration Manager
//* V1R1M0 5698-B66
...
//*
//* ****************************************************************
...
//KCIVARS DD *
ACTION           MIGRATE * CREATE|DISCOVER|GENERATE|DELETE|MIGRATE
RTE_NAME         DEMO
RTE_PLIB_HILEV   TEST1.TST


KFJ_SECURITY_EXITS_LIB  'TEST1.TST.DEMO.SECEXITS' * override security exits
library
```

The security exits will be migrated from legacy PARMGEN to the dataset specified in the KFJ_SECURITY_EXITS_LIB parameter.

The RTEDEF library will contain the runtime environment (RTE) definitions that are copied from the "old" RTE library in the MIGRATE step. After migration, you must review the RTEDEF member to verify that the RTE_X_SECURTY_EXITS_LIB statement lists the correct dataset name.

In addition, <*rte_plib_hilev*>.RTEDEF(<*rte_name*>) will contain a reference (parameter) to:

```
RTE_X_SECURTY_EXITS_LIB LEGACY.SECURITY.EXIT.LIBRARY
```

which points to the security exit library that was used in the legacy RTE. You may need to change this value to point to your custom library, rather than the default library.

Important: You must double check this value to verify it is correct before starting the GENERATE action.

Related reference

CREATE

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

Using embed overrides

With Monitoring Configuration Manager, it is possible to use embed overrides for the RTE that is being created. Two parameters, KFJ_USE_EMBEDS and KFJ_EMBEDS_LIB, can be specified in the JCL KCIVARS DD statement for CREATE and MIGRATE actions.

Embed override parameters

The embed override parameters are described below.

KFJ_USE_EMBEDS

Specify **Y** or **N** to indicate whether you want to use embedded override values for the RTE. The default is N.

A value of "Y" indicates to the respective MIGRATE or CREATE action that you will use *embed overrides* in the subject RTE.

When the parameter is set to "Y", Monitoring Configuration Manager will create an additional dataset next to the RTEDEF dataset containing the embed overrides for the products that are installed in the respective CSI used to build the RTE (in case of MIGRATE, this library will contain the embeds from the RTE being migrated from).

The default dataset is <*rte_plib_hilev*>.<*rte_name*>.EMBEDS, which allows you to isolate the Embed values into the respective libraries per RTE. The values in italics are for the high-level qualifier (HLQ) and the name of the RTE you want to use.

KFJ_EMBEDS_LIB

Specify the dataset containing the embedded override values. This parameter needs to point to a valid dataset name that is accessible by Monitoring Configuration Manager. An example of the dataset name is *highlevel*.PARMGEN.RTEDEF.EMBEDS.

> **Note:** This parameter is only active if KFJ_USE_EMBEDS is set to Y.

This parameter also allows you to override the standard embed override dataset name and instead point to a single dataset name, in case any common embed values are used for several RTEs.

If you will be using different override values for different RTEs, consider using the RTE_NAME or SYSNAME in the name of the embed library.

Example

An example of how to specify the *embed override* parameters is shown below. By default, the embed override support is disabled. However, this example shows that it is enabled and provides the dataset name.

```
//UID#ZMCM JOB ,CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=<lpar>
//* ********************************************************************
//* IBM Z Monitoring Configuration Manager
//* V1R1M0 5698-B66
...
//*
//* ********************************************************************
...
//KCIVARS DD *
ACTION           CREATE  * CREATE|DISCOVER|GENERATE|DELETE|MIGRATE
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
```

Troubleshooting

To troubleshoot issues with Monitoring 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 Monitoring 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
KFU00003I Workflow task recap: Programs=program_count MaxRC=max_program_rc
REXX=rexx_exec_count MaxRC=max_rexx_rc
KFU00004I KCIOMEGA is ending; RC=rc SYSPLEX=sysplex LPAR=lpar DATE=…
```

Figure 33. Example KCIPRINT output data set for a successful Monitoring Configuration Manager job

KCITRACE sysout data set

KCITRACE contains a superset of the KCIPRINT contents, including 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 Monitoring 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 numerous 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 Monitoring 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 Monitoring Configuration Manager contains *default* parameter values. It does *not* reflect the values in your RTEDEF library members. To see the parameter values used by Monitoring Configuration Manager, refer to the `>Parameter and symbol values` heading in the KCITRACE sysout data set.

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.

Review the report, correct the parameters, and then resubmit the job.

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.

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 dataset 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 dataset 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 dataset.

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 dataset 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 dataset, 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 dataset.

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 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 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 RTE_USS_RTEDIR directory *path*

Explanation

The **RTE_USS_RTEDIR** parameter specifies the path of the z/OS UNIX directory where the **GENERATE** action writes runtime members. The user who runs the job that performs the **GENERATE** action must be able to write to this z/OS UNIX directory.

> **Note:** PARMGEN, the configuration software that preceded IBM Z Monitoring Configuration Manager, requires z/OS UNIX superuser privileges. Write access is not an issue with PARMGEN, because superuser can write to *any* directory. By contrast, Monitoring Configuration Manager does *not* require superuser privileges. Instead, the user who runs Monitoring Configuration Manager must have the required privileges, such as write access to the **RTE_USS_RTEDIR** 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 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 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.

> **Note:** PARMGEN, the configuration software that preceded IBM Z Monitoring Configuration Manager, requires z/OS UNIX superuser privileges. Create permissions are not an issue with PARMGEN, because superuser can create a directory in *any* path. By contrast, Monitoring Configuration Manager does *not* require superuser privileges. Instead, the user who runs Monitoring Configuration Manager must have the required privileges, such as the permission to create write access to the **RTE_USS_RTEDIR** 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.

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
/*
```

KFJ00206W **RTE_SHARE *xxx* not supported. Changed to SMP.**

Explanation:
Monitoring 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

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=*count* MaxRC= *return code* REXX=*return code* MaxRC=*return code***

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 a previous error or warning message will indicate the cause of the problem and recommend corrective action.

KFU00004I KCIOMEGA is ending; RC=*return code* SYSPLEX=*name* LPAR=*name* DATE=*date and time*

Explanation

The KCIOMEGA workflow utility is ending. The job step return code, system, date and time are reported.

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 30.
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 will indicate the cause of the problem and recommend 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* 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:
None 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:
The z/OS MVS System Codes in IBM Knowledge Center 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 Knowledge Center 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 Knowledge Center for corrective action. Otherwise, contact IBM Software Support.

KFU00016E Program has failed; PROGRAM=*name* 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. This is likely to be the last output data set in the job.
- 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* 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:
None 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* 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. This is likely to be the last output data set in the job.
- 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.

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

$VALRPT sysout data set 85

A

Actions
    CREATE 27
    DELETE 34
    DISCOVER 28
    GENERATE 33
    MIGRATE 35
actions to run 26
authority required 5

B

Basic concept of product 1
batch interface 25
batch processing
    KCIOMEGA 36

C

commands overview 1
comparison with legacy PARMGEN 2
CREATE action 26, 27
creating runtime environment 7

D

defining OMEGAMON subsystem 6
definition library for RTE
    initial members 63
    members 61
    order of members 62
DELETE action 26, 34
DISCOVER action 26, 28

E

embed override 83
exits
    security 81

G

GENERATE
    replaces maintenance scenarios 37
GENERATE action
    location of runtime members 69

I

interface
interface *(continued)*
    batch 25

J

JCL to run product 25

K

KCIOMEGA workflow 36
KCIPRINT sysout data set 85
KCITRACE sysout data set 85
KCIVARS 25
KFJ messages 87
KFJ_EMBEDS_LIB parameter 83
KFJ_SYSNAME parameter 55
KFJ_USE_EMBEDS parameter 83
KFU messages 91

M

maintenance scenarios vs. GENERATE 37
messages
    KFJ messages 87
    KFU messages 91
MIGRATE action 26, 35
monitoring components 73

O

OMEGAMON subsystem
    defining 6
override embedded value 83

P

Parameters
    different default values 56
    for communication between servers 73
    initial RTE 39
    KFJ_EMBEDS_LIB 83
    KFJ_SYSNAME 55
    KFJ_USE_EMBEDS 83
    new 49
    significant default values 49
    spare parm tables 57
    SYSNAME 55
    use of variables 79
    validation report 85
prerequisites for product 5
privileges to access product 5

R

RTE_NAME 25
RTE_PLIB_HILEV 25

RTEDEF
members for variables 79
runtime environment
basic extra parameters 49
creating 7
creating or updating 14
definition library 61
definition library members 61
different default parameters 56
in a sysplex 73
initial library members 63
new parameters 49
order of definition library members 62
parameters 39
RTE member locations 69
sparse parameters tables 57

S

security exits 81
SYSNAME parameter 55
SYSPRINT sysout data set 85

T

troubleshooting 85

U

updating runtime environment 14

V

variables
in parameter values 79

Product Number: 5698-B66

SC28-3147