IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS
Version 6.2.3 Fix Pack 1

Common Planning and Configuration Guide

IBM
IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS
Version 6.2.3 Fix Pack 1

Common Planning and Configuration Guide
Note

Before using this information and the product it supports, read the information in "Notices" on page 213.

This edition applies to version 6, release 2, modification 3, Fix Pack 1 of IBM Tivoli Management Services on z/OS (product number 5698-A79) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC23-9734-03.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Figures ........................................ ix
Tables ........................................... xi

Part 1. Planning your monitoring environment ........................................ 1

Chapter 1. Overview of the OMEGAMON XE products and IBM Tivoli Management Services on z/OS .................................................. 3
Tivoli Management Services and OMEGAMON XE components ...................... 3
  Tivoli Enterprise Monitoring Server ........................................ 4
  IBM Tivoli OMEGAMON enhanced 3270 user interface ................................ 4
  Monitoring agents ........................................................................ 4
  Tivoli Enterprise Portal Server and clients ..................................... 5
  Tivoli Data Warehouse ................................................................. 5
  TMS:Engine ............................................................................. 6
  Event synchronization component ............................................. 6
  Tivoli Enterprise Portal Server extended services ................................ 6
OMEGAMON XE shared components ..................................................... 6
Interoperability and integration with other products ...................................... 7
New in this release ........................................................................... 8
  IBM Tivoli OMEGAMON enhanced 3270 user interface ....................... 8
  PARMGEN configuration method .................................................. 8
  Support for self-describing agents ............................................... 8
  High-availability hub .................................................................. 9
  Automatic registration of products with the local monitoring server .............. 9
  Tivoli Enterprise Monitoring Server support for audit logging .............. 9
  Web Services SOAP Server available on remote Tivoli Enterprise Monitoring Server ................................................................. 9
Authorization of z/OS Take Action commands by NetView .......................... 10

Chapter 2. Prerequisites and packaging ................................................... 11
Software and hardware prerequisites ..................................................... 11
Packaging ....................................................................................... 11

Chapter 3. Planning your deployment ..................................................... 15
Decision 1: Whether to install into a shared CSI ..................................... 15
Decision 2: Where to install your hub and remote monitoring servers .......... 16
  Deciding where to place the hub monitoring server ................................ 16
  Self-describing agents. ................................................................ 17
  High-availability hub monitoring server ....................................... 18
  Deciding where to place remote monitoring servers .......................... 19
  Monitoring server names .................................................................. 20
  Planning for the SOAP server on z/OS systems ............................. 20
Decision 3: Where to configure your monitoring agents .......................... 22
  Self-describing agents. ................................................................ 23
  Autonomous agents .................................................................... 23
Decision 4: Which user interface to use .............................................. 23
  IBM Tivoli OMEGAMON enhanced 3270 user interface .................. 24
  OMEGAMON (3270) and OMEGAMON II (CUA) interfaces ............. 24
  Tivoli Enterprise Portal interface .................................................. 24
Decision 5: Which configuration method to use .................................... 25
  Configuration parameters ......................................................... 26
  System variables ....................................................................... 26
Decision 6: How to set up communications between components .................. 27
Step 1. Create or update the Configuration Tool work library. ........................................ 77
  If you use a cloned copy of a CSI containing the Configuration Tool ......................... 77
  If you use a new CSI ............................................................................................................ 78
Step 2. Start the Configuration Tool .................................................................................. 78
Step 3. Set up the Configuration Tool work environment ............................................... 79
Step 4. Set up and build a runtime environment ............................................................... 83
Step 5. Configure your products and components in the runtime environment ............... 95
  Configure a monitoring server ......................................................................................... 95
  Set appropriate parameter values .................................................................................. 96
  Verify the configuration before loading the runtime libraries ..................................... 100
  Load runtime libraries once for multiple products ....................................................... 100
  Eliminate duplicate Complete the Configuration steps .............................................. 101

Chapter 6. Completing the configuration ....................................................................... 103
Add support for the SYSTCPD DDNAME in the started tasks ..................................... 103
Copy started task procedures to your procedure library .............................................. 103
Copy the VTAM definitions to your system VTAMLST ............................................... 104
Vary the VTAM major node active .................................................................................. 105
APF-authorize the runtime load libraries ....................................................................... 105
Complete the configuration for the OMEGAMON subsystem (optional) .................... 106
  Update the IEFSSNxx member of the SYS1.PARMLIB .................................................. 106
  Copy the OMEGAMON subsystem started task ............................................................. 106
  Update the LINKLIST ........................................................................................................ 106
  Use the RESTART parameter ....................................................................................... 106
  Verify the OMEGAMON subsystem installation ............................................................ 107
Complete the configuration for OMEGAMON monitoring agents ............................. 107
Enable the self-describing agent feature or manually install application support ............ 107
Install language support .................................................................................................... 108
Verify the configuration ..................................................................................................... 108
Enable security .................................................................................................................. 111
  Enabling SNMP V3 passwords for autonomous agents ................................................. 112
Enable security on the IBM Tivoli OMEGAMON enhanced 3270 user interface ......... 112
  Define an SAF general resource class ........................................................................... 113
  Define LOGON profiles to control access to the interface ............................................ 114
  Define Take Action profiles to control access to data actions ...................................... 114
  Define Query profiles to control access to data sources .............................................. 115
  Permit access to profiles ............................................................................................... 115
  Define a PASSTKT profile ............................................................................................. 116
Enable historical data collection (optional) ................................................................... 116
  Provide access to the persistent data store files ............................................................. 116
  Authorize the KPDDSCO module ................................................................................... 116
  Verify persistent data store configuration ........................................................................ 117
  Enable historical data collection within the Tivoli Enterprise Portal ......................... 117
Run the ITMSUPER Tools (optional) ................................................................................ 117

Part 3. Configuration Tool (ICAT) scenarios ................................................................... 119

Chapter 7. Using the Configuration Tool (ICAT) batch mode to replicate a configured
  environment ....................................................................................................................... 121
Step 1. Create the batch mode job ..................................................................................... 122
  Creating the batch mode job ......................................................................................... 122
  Batch mode job submission options ............................................................................. 123
  Batch mode configuration jobs ...................................................................................... 124
  Batch mode job report .................................................................................................... 124
Step 2. Create and clone a batch mode parameter member ........................................... 125
Step 3. Adjust and validate the parameters ...................................................................... 128
Step 4. Generate sample transport JCL .................................. 130
Step 5. Create a new runtime environment .................................. 131
Scenario A: Defining a runtime environment on a z/OS image using shared DASD .... 133
Scenario B: Transporting a runtime environment from one z/OS image to another ... 133
Scenario C: Transporting runtime environment batch jobs from one z/OS image to another equipped with the Configuration Tool ................. 134
Scenario D: Transporting runtime environment batch mode parameters from one z/OS image to another .................................. 136
Step 6. Complete the configuration of a new runtime environment ..................... 137

Chapter 8. Configuration Tool (ICAT) deployment scenarios ......................... 139
Scenario 1: Standard replication method ...................................... 139
Scenario 2: Using common RKANPARU and RKANCMGU libraries ................. 141
Scenario 3: Using batch mode to replicate a runtime environment to another LPAR . 143
Scenario 4: Using batch auto-submit to replicate a runtime environment to another LPAR .... 144
Scenario 5: Using batch mode to replicate a runtime environment on the same LPAR .... 146
Scenario 6: Using interactive copy to replicate a runtime environment ............... 147
Scenario 7: Copying configuration values from one instance to another of the Configuration Tool 147
Scenario 8: Configuring a remote monitoring server and monitoring agents in batch mode 148
Configuration steps ........................................................... 149
Step 1. Create the batch mode job (CICATB) .................................. 149
Step 2. Clone the runtime environment batch parameter member ..................... 150
Step 3. Review and update the parameter member for the new runtime environment . 153
Step 4. Run the CICATB job to validate the changes ................................ 156
Step 5. Create and (optionally) execute the batch jobs to create the runtime environment 157
Step 6. Customize and complete the configuration .................................. 162

Appendix A. Enabling system variable support .................................. 165
Using the PARMGEN Workflow user interface to set up system variables ............... 165
Prerequisites ................................................................. 165
Enable system variable support .............................................. 166
Specify user-defined symbols and variables ...................................... 167
Using the Configuration Tool (ICAT) to set up system variables ...................... 167
Enable system variable support .............................................. 168
Create the system variable parameter member .................................... 169
Create the VTAM major node rename job .................................... 170
Create one VTAM major node for all products in the runtime environment .......... 170

Appendix B. Configuration Tool reference ....................................... 173
Configuration services and utilities ............................................ 173
Runtime environment utilities ................................................ 174
Batch mode utilities .......................................................... 178
KCISETP: Set up the environment for batch utilities ................................ 178
KCIICFKEY: Manage ISPF PF key .......................................... 179
KCIICPGHP: Display help for configuration parameters .......................... 180
The Runtime Environments (RTE) panel .................................... 180
Loading runtime environments ............................................... 182
Deleting a runtime environment ............................................. 183

Appendix C. Maintaining the persistent data store ................................... 185
Overview of the persistent data store on z/OS .................................. 185
Automatic maintenance process ................................................ 186
The part of maintenance you control ......................................... 187
Indicating data set backup to tape or to DASD ................................ 187
Naming the export data sets .................................................. 187
Making archived data available ................................................ 188
Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Agent/server/client architecture</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>High-availability runtime environment</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Full runtime environment on a single system</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>Full runtime environments on several systems</td>
<td>39</td>
</tr>
<tr>
<td>5</td>
<td>Base runtime environment</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>Sharing-with-base runtime environment</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>Sharing-with-full runtime environment</td>
<td>43</td>
</tr>
<tr>
<td>8</td>
<td>Sharing-with-SMP/E runtime environment</td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td>Welcome panel</td>
<td>56</td>
</tr>
<tr>
<td>10</td>
<td>Set up the PARMGEN work environment for a runtime environment (1 of 3)</td>
<td>58</td>
</tr>
<tr>
<td>11</td>
<td>Set up the PARMGEN work environment for a runtime environment (2 of 3)</td>
<td>59</td>
</tr>
<tr>
<td>12</td>
<td>Set up the PARMGEN work environment for a runtime environment (3 of 3)</td>
<td>60</td>
</tr>
<tr>
<td>13</td>
<td>Display PARMGEN environment analysis</td>
<td>61</td>
</tr>
<tr>
<td>14</td>
<td>Exclude products from PARMGEN customization</td>
<td>62</td>
</tr>
<tr>
<td>15</td>
<td>Customize PARMGEN configuration profile members</td>
<td>64</td>
</tr>
<tr>
<td>16</td>
<td>Update the user prolog</td>
<td>65</td>
</tr>
<tr>
<td>17</td>
<td>Verify the list of products in the runtime environment.</td>
<td>66</td>
</tr>
<tr>
<td>18</td>
<td>Update names of started tasks</td>
<td>66</td>
</tr>
<tr>
<td>19</td>
<td>Update TCP/IP port values</td>
<td>67</td>
</tr>
<tr>
<td>20</td>
<td>Update VTAM node and application definitions</td>
<td>68</td>
</tr>
<tr>
<td>21</td>
<td>Enable APF-authorization statements</td>
<td>69</td>
</tr>
<tr>
<td>22</td>
<td>Enable the self-describing agent feature</td>
<td>69</td>
</tr>
<tr>
<td>23</td>
<td>Update SMS-related values</td>
<td>70</td>
</tr>
<tr>
<td>24</td>
<td>Configure an OMEGAMON enhanced 3270 user interface</td>
<td>70</td>
</tr>
<tr>
<td>25</td>
<td>Create the runtime environment members and jobs</td>
<td>73</td>
</tr>
<tr>
<td>26</td>
<td>Submit batch jobs to complete PARMGEN setup</td>
<td>74</td>
</tr>
<tr>
<td>27</td>
<td>Configuration Tool Main Menu</td>
<td>79</td>
</tr>
<tr>
<td>28</td>
<td>Set up Work Environment panel</td>
<td>80</td>
</tr>
<tr>
<td>29</td>
<td>Specify JCL Options panel</td>
<td>80</td>
</tr>
<tr>
<td>30</td>
<td>Set up Configuration Environment panel</td>
<td>82</td>
</tr>
<tr>
<td>31</td>
<td>Configure Products panel</td>
<td>84</td>
</tr>
<tr>
<td>32</td>
<td>Product Selection Menu</td>
<td>84</td>
</tr>
<tr>
<td>33</td>
<td>Runtime Environments (RTEs) panel</td>
<td>85</td>
</tr>
<tr>
<td>34</td>
<td>Add Runtime Environment panel for a base runtime environment</td>
<td>86</td>
</tr>
<tr>
<td>35</td>
<td>Add Runtime Environment (1 of 3) panel for a full or sharing runtime environment.</td>
<td>87</td>
</tr>
<tr>
<td>36</td>
<td>Add Runtime Environment (2 of 3) panel for a full or sharing runtime environment.</td>
<td>90</td>
</tr>
<tr>
<td>37</td>
<td>Add Runtime Environment (3 of 3) panel for a full or sharing runtime environment.</td>
<td>93</td>
</tr>
<tr>
<td>38</td>
<td>Verify Configuration and Generate Parameter Map panel</td>
<td>96</td>
</tr>
<tr>
<td>39</td>
<td>Specify Nonstandard Parameters panel</td>
<td>99</td>
</tr>
<tr>
<td>40</td>
<td>Create/Clone Batch Mode Parameters panel</td>
<td>126</td>
</tr>
<tr>
<td>41</td>
<td>SYSTSPRT for successful scan</td>
<td>129</td>
</tr>
<tr>
<td>42</td>
<td>Batch scenario configuration: z/OS remote monitoring server reporting to z/OS hub</td>
<td>148</td>
</tr>
<tr>
<td>43</td>
<td>Runtime Environments (RTEs) panel</td>
<td>151</td>
</tr>
<tr>
<td>44</td>
<td>RTE Utility Menu panel</td>
<td>151</td>
</tr>
<tr>
<td>45</td>
<td>Create/Clone Parameter Deck Menu panel</td>
<td>152</td>
</tr>
<tr>
<td>46</td>
<td>Create/Clone Batch Mode Parameters panel</td>
<td>152</td>
</tr>
<tr>
<td>47</td>
<td>Batch parameter member for SYSB</td>
<td>154</td>
</tr>
<tr>
<td>48</td>
<td>SYSLOG/OPERLOG messages</td>
<td>162</td>
</tr>
<tr>
<td>49</td>
<td>Runtime Environments (RTEs) panel with new SYSB runtime environment</td>
<td>162</td>
</tr>
<tr>
<td>50</td>
<td>Pre-defined and user-defined symbolics</td>
<td>167</td>
</tr>
<tr>
<td>51</td>
<td>Configuration Services and Utilities panel</td>
<td>173</td>
</tr>
<tr>
<td>52</td>
<td>RTE Utility Menu panel</td>
<td>175</td>
</tr>
<tr>
<td>53</td>
<td>Runtime environments panel</td>
<td>180</td>
</tr>
<tr>
<td>No.</td>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>54.</td>
<td>README Table of Contents Display panel</td>
<td>182</td>
</tr>
<tr>
<td>55.</td>
<td>Delete runtime environment panel</td>
<td>183</td>
</tr>
</tbody>
</table>
### Tables

1. Common components of OMEGAMON XE monitoring agents on z/OS ........................................ 6  
2. Product packaging, Tivoli Management Services on z/OS ..................................................... 13  
3. Product packaging, OMEGAMON XE monitoring agents ....................................................... 14  
4. Configuration methods ............................................................................................................. 18  
5. Types of libraries ....................................................................................................................... 35  
6. Runtime environment data set naming convention .................................................................. 35  
7. Types of runtime environments ............................................................................................... 37  
8. Validation of the runtime environment ................................................................................... 109  
9. Data validation on user interfaces .......................................................................................... 111  
10. CICATB job reports ............................................................................................................... 125  
11. Runtime environment transport methods ................................................................................ 132  
12. Thumbnail configuration scenario 1: standard replication method ........................................... 139  
13. Thumbnail configuration scenario 2: Using common RKANPARU and RKANCMDU libraries ... 141  
14. Thumbnail configuration scenario 3: Using batch mode to replicate a runtime environment to another LPAR ................................................................................................................. 143  
15. Thumbnail configuration scenario 4: Using batch auto-submit to replicate a runtime environment to another LPAR ................................................................................................................. 144  
16. Thumbnail configuration scenario 5: Using batch mode to replicate a runtime environment on the same LPAR ............................................................................................................. 146  
17. Thumbnail configuration scenario 6: Using interactive copy to replicate a runtime environment .......................................................... 147  
18. Thumbnail configuration scenario 7: Copying configuration values from one instance to another of the Configuration Tool ........................................................................................................... 148  
19. Actions required to update the runtime environment parameters .......................................... 154  
20. Batch parameter format .......................................................................................................... 180  
21. Actions required for configuration and maintenance tasks ..................................................... 182  
22. Section 1 data record format, exported persistent data ............................................................ 191  
23. Section 2 data record format, exported persistent data ............................................................ 192  
24. Section 2 table description record, exported persistent data .................................................... 192  
25. Section 2 column description record, exported persistent data ................................................ 192  
26. Section 3 record format, exported persistent data .................................................................... 193  
27. Two-character product codes ................................................................................................ 201  
28. OMEGAMON started tasks and automation dependencies .................................................... 203  
29. OMEGAMON started tasks by priority group ........................................................................ 205
Part 1. Planning your monitoring environment

The chapters in this part of the Common Planning and Configuration Guide provide the information required for beginning to configure the common components and monitoring agents on z/OS®.

- Chapter 1, “Overview of the OMEGAMON XE products and IBM Tivoli Management Services on z/OS,” on page 3 explains the architecture of the commonly shared components of IBM® Tivoli® Management Services and of the OMEGAMON® XE monitoring agents. Chapter 1 also contains a New in this release section, which describes changes and enhancements that affect the configuration of the components on z/OS.

- Chapter 2, “Prerequisites and packaging,” on page 11 provides information about hardware and software prerequisites, packaging, and special requirements for communications and security.

- Chapter 3, “Planning your deployment,” on page 15 helps you plan for installing and configuring the common components and OMEGAMON XE monitoring agents on z/OS. Chapter 3 covers some basic planning decisions:
  
  - “Decision 1: Whether to install into a shared CSI” on page 15
  - “Decision 2: Where to install your hub and remote monitoring servers” on page 16
  - “Decision 3: Where to configure your monitoring agents” on page 22
  - “Decision 4: Which user interface to use” on page 23
  - “Decision 5: Which configuration method to use” on page 25
  - “Decision 6: How to set up communications between components” on page 27
  - “Decision 7: What types of runtime environments to set up” on page 35
  - “Decision 8: What historical data to collect and how to manage it” on page 45
  - “Decision 9: Which security options to enable” on page 48
Chapter 1. Overview of the OMEGAMON XE products and IBM Tivoli Management Services on z/OS

The OMEGAMON XE products are a suite of products used to monitor and manage sophisticated mainframe applications and environments. The OMEGAMON XE products share a common technology, Tivoli Management Services, to provide security, data transfer and storage, notification, user interface presentation, and communication for a number of Tivoli monitoring products.

This guide describes the planning and configuration procedures common to the OMEGAMON XE products and the shared components that run on z/OS. For instructions on configuring a specific OMEGAMON XE product, see the configuration guide for that product after you have completed the steps described in this guide. For instructions on performing the SMP/E installation of the products and components, see IBM Tivoli Management Services on z/OS: Program Directory for IBM Tivoli Management Services on z/OS and see the Program Directory for each OMEGAMON XE product.

Note: In the documentation for the Tivoli Management Services on z/OS and OMEGAMON XE products, installation includes the SMP/E installation only, and configuration includes the remaining steps to set up the product to run successfully in your environment.

Tivoli Management Services and OMEGAMON XE components

The OMEGAMON XE suite of products uses the Tivoli Management Services infrastructure, which provides security, data transfer and storage, notification mechanisms, user interface presentation, and communication services for products in the IBM Tivoli Monitoring and IBM Tivoli OMEGAMON XE suites in an agent/server/client architecture. Figure 1 shows a simple topology of the various principal components that form the Tivoli Management Services and OMEGAMON XE monitoring infrastructure, and how they interact with each other.

![Figure 1. Agent/server/client architecture](image-url)
Note: Not all the Tivoli Management Services and OMEGAMON XE components in Figure 1 on page 3 are required for all customer environments. For example, you do not have to have a Tivoli Enterprise Portal Server installed in your environment to run OMEGAMON XE on z/OS and use the IBM Tivoli OMEGAMON enhanced 3270 user interface to view data. However, if you enable the self-describing agent feature provided by Tivoli Management Services, the Tivoli Enterprise Services user interface extension is required to control the administration of the self-describing agent feature. This component provides the tacmd command-line utility to interact with a monitoring server. This component is available on the IBM Tivoli Monitoring DVD image.

Tivoli Enterprise Monitoring Server
Tivoli Enterprise Monitoring Server (also called the monitoring server) is the nerve center of Tivoli Management Services. The monitoring server performs the following tasks:

- Consolidates the data collected by monitoring agents and distributes the data to the connected clients (OMEGAMON enhanced 3270 user interface and Tivoli Enterprise Portal Server).
- Manages the connection status of the monitoring agents.
- Receives commands from the connected clients and distributes them to the appropriate monitoring agents.
- Sends alerts to the connected clients when specified availability and performance problems are detected.
- Stores historical data and configuration prototypes.

The master monitoring server is called the hub monitoring server. The hub monitoring server acts as the focal point for data collection and distribution. It communicates with monitoring agents, with the OMEGAMON enhanced 3270 user interface, with the Tivoli Enterprise Portal Server, and with the Warehouse Proxy and Summarization and Pruning agents (see “Tivoli Data Warehouse” on page 5).

A remote monitoring server is remote only with respect to the hub monitoring server, not with respect to the monitoring agents. Remote monitoring servers communicate only with the monitoring agents that report to them and with the hub monitoring server to which the monitoring servers report.

Monitoring servers can run on z/OS, Windows, AIX®, HP-UX, Solaris, or Linux systems.

IBM Tivoli OMEGAMON enhanced 3270 user interface
The IBM Tivoli OMEGAMON enhanced 3270 user interface component is the latest generation of 3270 interfaces for the OMEGAMON family of monitoring products. The new interface preserves many of the valued features of the earlier 3270-based interfaces, but extends the functionality to include many new features including cross-system and cross-product reporting.

The OMEGAMON enhanced 3270 user interface component enables you to monitor the performance of your z/OS systems, applications, and devices in your environment and helps you identify and troubleshoot problems with those monitored resources. In the interface, data is presented in workspaces and subpanels in which the collected data and relevant information is displayed. You can quickly and easily diagnose problems with monitored resources and take action to correct them. You can also customize the workspaces to suit your requirements, and even design and create your own workspaces and navigation.

For more information, see the IBM Tivoli OMEGAMON XE and Tivoli Management Services on z/OS: Enhanced 3270 User Interface Guide.

Monitoring agents
Monitoring agent products are available for systems, database products, and applications. You can see the complete list of monitoring agents at the Service Availability and Performance Management website: http://www.ibm.com/software/tivoli/solutions/availability/products.html
OMETAMON XE monitoring agents are located on monitored, or managed, systems. The agents pass the system or application data they collect to a Tivoli Enterprise Monitoring Server, and the data is passed to a connected client user interface (OMETAMON enhanced 3270 user interface or the Tivoli Enterprise Portal Server and displayed in a Tivoli Enterprise Portal client). Monitoring agents can also sample data at specified intervals and store it for short-term historical collection, which can then be transferred to a Tivoli Data Warehouse. Finally, the monitoring agents can also compare the current values of monitored properties against a set of defined conditions and thresholds, and trigger alerts or actions when the current values match the defined conditions or surpass the thresholds.

OMETAMON XE monitoring agents are available for the following products:

- IBM Tivoli OMETAMON XE for CICS® on z/OS
- IBM Tivoli OMETAMON XE for DB2® Performance Expert/Performance Monitor on z/OS
- IBM Tivoli OMETAMON XE for IMS on z/OS
- IBM Tivoli OMETAMON XE for Mainframe Networks on z/OS
- IBM Tivoli OMETAMON XE for Storage on z/OS
- IBM Tivoli OMETAMON XE on z/OS
- IBM Tivoli OMETAMON XE for Messaging on z/OS
- IBM Tivoli OMETAMON XE for z/VM® and Linux

Tivoli Enterprise Portal Server and clients

Tivoli Enterprise Portal (also called the portal or the portal client) is the user interface for products using Tivoli Management Services. The Tivoli Enterprise Portal is a thin Java™ client application. It has its own server, the Tivoli Enterprise Portal Server, that communicates with the hub monitoring server to send requests to and retrieve data from monitoring agents on managed systems. Tivoli Enterprise Portal Server (the portal server) builds and formats the portal workspaces that display real-time and historical data collected by the monitoring agents. The portal server can run on Windows, AIX, or Linux systems.

You can access the portal client in any of the following ways:

- Browser client (Internet Explorer or Mozilla Firefox on Windows, Linux, or AIX) connected to a web server embedded in the portal server.
- Desktop client installed on a Windows or Linux system.
- Desktop client downloaded and run by IBM Web Start for Java, and updated at every startup.

For setup information about the portal server and client, see the IBM Tivoli Monitoring: Installation and Setup Guide.

Tivoli Data Warehouse

Tivoli Data Warehouse, an optional component of Tivoli Management Services, is a long-term data store for the performance and analysis data collected by the monitoring agents. The Warehouse Proxy agent periodically receives data from the hub monitoring server or from the monitoring agents and inserts the data into the Tivoli Data Warehouse. On a z/OS system, short-term history data for monitoring agents is maintained in data sets allocated and initialized during product configuration. The Warehouse Proxy agent receives the short-term history data and delivers it to the warehouse.

Two specialized agents interact with the Tivoli Data Warehouse:

- The Warehouse Proxy agent receives the short-term history data and delivers it to the Tivoli Data Warehouse.
- You can use the Summarization and Pruning agent to customize how long to save (pruning) and how often to aggregate (summarization) the data in the Tivoli Data Warehouse database.
The Tivoli Data Warehouse, the Warehouse Proxy agent, and the Summarization and Pruning agent can run on Windows, AIX, HP-UX, Solaris, or Linux systems. The Tivoli Data Warehouse can also run on z/OS. For more information, see *IBM Tivoli Monitoring: Installation and Setup Guide* and *IBM Tivoli Monitoring: Administrator's Guide*.

**TMS:Engine**

TMS:Engine provides common functions such as communications, multithreaded runtime services, tracing, and logging for the monitoring server and monitoring agents on z/OS. This shared component enables common portable code to make platform-independent system calls. This allows Tivoli Enterprise Monitoring Server code to be compiled for and executed on z/OS, Windows, Linux, and UNIX platforms.

**Event synchronization component**

The event synchronization component sends updates to situation events that have been forwarded to a Tivoli Enterprise Console® event server or an IBM Tivoli Netcool®/OMNIbus ObjectServer back to the monitoring server. In the Tivoli Enterprise Portal, the Situation Event Console, the Common Event Console and the Tivoli Enterprise Console event views are synchronized with the updated status of the events.

For information about the various configurations of monitoring servers and event servers that you can have in your environment, see the *IBM Tivoli Monitoring: Installation and Setup Guide*.

You can set up stand-alone monitoring agents (those that are configured in their own address spaces) to run in autonomous mode (without communicating directly with a monitoring server). An autonomous agent can emit Simple Network Management Protocol (SNMP) traps and Event Integration Facility (EIF) events directly to a Netcool/OMNIbus ObjectServer for agent-specific situations (but not for enterprise situations).

The *IBM Tivoli Monitoring: Installation and Setup Guide* provides instructions for configuring Netcool/OMNIbus ObjectServers to receive the events. For information on specifying which situation events to forward, see the Tivoli Enterprise Portal online help and the *IBM Tivoli Monitoring: Tivoli Enterprise Portal User’s Guide*. For detailed information about configuring and managing autonomous agents, see the "Agent autonomy" chapter of the *IBM Tivoli Monitoring: Administrator’s Guide*.

**Tivoli Enterprise Portal Server extended services**

Tivoli Enterprise Portal Server extended services (TEPS/e) is an embedded, co-located extension of the Tivoli Enterprise Portal Server that provides J2EE-based Application Server integration facilities. TEPS/e provides support for a federated user repository. For more information about TEPS/e, see the *IBM Tivoli Monitoring: Administrator’s Guide*.

**OMEGAMON XE shared components**

OMEGAMON XE monitoring agents on z/OS share several common components, which are included on the product tape (see Table 1). If you install into an existing environment, with the components at the required level, you might have to delete the FMIDs for these components from the SMP/E installation jobs to avoid errors because the components are already installed. See the *Program Directory* for each product for more information.

<table>
<thead>
<tr>
<th><strong>Table 1. Common components of OMEGAMON XE monitoring agents on z/OS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OMEGAMON subsystem</strong></td>
</tr>
<tr>
<td><strong>End-to-End Response Time collector</strong></td>
</tr>
</tbody>
</table>
### Interoperability and integration with other products

The OMEGAMON XE family of monitoring agents are designed to integrate with each other and with other products that use Tivoli Management Services. These products exploit the power of the Tivoli Enterprise Portal to integrate and correlate performance and availability information from a variety of sources. For example, you can create context-sensitive links between product workspaces to obtain additional information about systems, subsystems, resources, or network components that are being monitored by other monitoring agents. You can also create links from product workspaces to TN3270-based applications.

Additionally, if you have purchased IBM Tivoli OMEGAMON DE on z/OS, you can create custom Tivoli Enterprise Portal workspaces composed of data from a range of Tivoli monitoring solutions (IBM Tivoli Monitoring, IBM Tivoli Composite Application Management, IBM Tivoli NetView® for z/OS, and OMEGAMON XE products). You can use OMEGAMON DE to display metrics from operating systems, middleware, databases, storage, web application servers, and network sources for integrated performance views on a single screen.

OMEGAMON DE on z/OS adds the Status Data Manager bridge component of OMEGAVIEW II for the Enterprise to your Tivoli Management Services environment. After you install OMEGAMON DE on z/OS, you can set up Tivoli Enterprise Portal workspaces that display data collected by more than one OMEGAMON XE product or that integrate OMEGAMON XE data with data from other products that use the Tivoli Enterprise Portal. You can create a consolidated situation for a combination of products. You can also use OMEGAMON DE on z/OS to set up advanced automation policies for notification and resolution of complex problems. For further information about the components and capabilities of OMEGAMON DE on z/OS, see [IBM Tivoli OMEGAMON DE on z/OS documentation](#).

Additionally, OMEGAMON XE products are being integrated with an increasing number of other IBM products. You can generate reports using Tivoli Common Reporting, which gathers data collected by OMEGAMON XE products and presents results in a format allowing the analysis of trends and key metrics in a consistent and integrated manner. Situation events reported by OMEGAMON XE monitoring agents can be forwarded to Tivoli Event Console or IBM Tivoli Netcool/OMNibus for event correlation and management. From the Tivoli Enterprise Portal, you can launch into other web-based or web-enabled Tivoli applications without having to re-enter user credentials, and you can launch in context into the Tivoli Enterprise Portal from applications like Tivoli Business Services Management.

For more information on product integration, see the user's guides for your products.

---

Table 1. Common components of OMEGAMON XE monitoring agents on z/OS (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMNIMON base (includes the IBM Tivoli OMEGAMON enhanced 3270 user interface)</td>
<td>A set of common code used by several OMEGAMON products to control initialization, security, and I/O for all sessions.</td>
</tr>
<tr>
<td>Shared probes</td>
<td>Data probes shared by several OMEGAMON XE products.</td>
</tr>
</tbody>
</table>
New in this release

Recent enhancements affect the configuration of Tivoli Management Services on z/OS and OMEGAMON XE monitoring agents. You might already be familiar with some of the changes described here if you have upgraded your monitoring environment. Each OMEGAMON XE monitoring agent also provides a number of additional new features. See the product-specific Planning and Configuration guides for further details.

**IBM Tivoli OMEGAMON enhanced 3270 user interface**

The IBM Tivoli OMEGAMON enhanced 3270 user interface component is the latest generation of 3270 interfaces for the OMEGAMON family of monitoring products. The new interface preserves many of the valued features of the earlier 3270-based interfaces, but extends the functionality to include many new features including cross-system and cross-product reporting.

As it is a shared OMEGAMON XE component, only a single instance of the OMEGAMON enhanced 3270 user interface address space is required within the customer sysplex to provide a client for users of all supported OMEGAMON XE products. The products that support the OMEGAMON enhanced 3270 user interface provide predefined workspaces that enable you to quickly and easily diagnose problems with monitored resources and take action to correct them. You can customize the workspaces to suit your requirements, and even design and create your own workspaces and navigation.

**Note:** Not all OMEGAMON XE agents currently provide support for the OMEGAMON enhanced 3270 user interface. See the product-specific Planning and Configuration guides for confirmation of support.

For more information, see the [IBM Tivoli OMEGAMON XE and Tivoli Management Services on z/OS: OMEGAMON Enhanced 3270 User Interface Guide](#).

**PARMGEN configuration method**

The PARMGEN configuration method is now the preferred method of product configuration. It is particularly suitable for new OMEGAMON XE customers or existing OMEGAMON XE customers in environments in which one person is responsible for configuring all OMEGAMON XE products and components.

By using the PARMGEN Workflow user interface, you edit a comprehensive list of parameters for configuring all installed products and components. You then submit a series of jobs to create a complete runtime environment with the parameter values you specified. If you have runtime environments that are already configured by the Configuration Tool (ICAT) method, a conversion tool is provided to enable you to use the existing parameter values in those runtime environments to set up initial values for new runtime environments to be configured by the PARMGEN method.

**Note:** The initial version of the PARMGEN configuration method provided in previous releases of Tivoli Management Services was known as PARMLIB.

For more information, see [Chapter 4, “Configuring products using the PARMGEN Workflow user interface,”](#) on page 55, and the [IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Common Parameter Reference](#).

**Support for self-describing agents**

In earlier releases, the initial installation and version upgrades required you to add application support for your monitoring agents (also known as 'seeding') to the Tivoli Enterprise Monitoring Server for the agents to function correctly. The Tivoli Enterprise Monitoring Server and OMEGAMON XE monitoring agents now provide self-describing agent support. After startup, self-describing agents automatically push application support updates to the self-describing hub monitoring server, the Tivoli Enterprise Portal Server, and the Tivoli Enterprise Portal browser client.
Self-describing monitoring agents eliminate the requirement to recycle the monitoring server after application support updates, and ensure that application support files are current. By default, the self-describing agent feature is enabled within any remote monitoring server and any monitoring agent that provides self-describing support but the feature is disabled at the hub monitoring server. If your enterprise includes self-describing agents, you can enable the self-describing agent feature by changing parameter values for the hub monitoring server.

**Note:** Not all OMEGAMON XE agents currently provide self-describing agent support. See the product-specific Planning and Configuration guides for confirmation of support.

For more information, see *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.*

**High-availability hub**

You can now configure a high-availability hub monitoring server in any Sysplex environment with dynamic virtual IP addressing (DVIPA) and shared DASD. A high-availability hub is configured in its own runtime environment, without any monitoring agents, and configurable on the same LPAR with a remote monitoring server. This configuration allows the hub monitoring server to be relocated to any suitable LPAR in the Sysplex with no changes, and with minimal disruption to the components connecting to the hub.

For instructions on configuring a high-availability hub on the z/OS operating system, see *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.* For detailed information about the high-availability hub on z/OS, including information about conversion from a static hub to a high-availability hub, see *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.* For a scenario, see *IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: PARMGEN Reference.*

**Automatic registration of products with the local monitoring server**

The runtime environment load processing option now automatically copies the product attribute and catalog members from the &thilev.TKANDATV target data set that was installed by SMP/E to the &rhilev.&rte.RKANDATV runtime environment. You do not have to regenerate and rerun the "Register with local TEMS" job for each product in the runtime environment if you apply maintenance that updates the product attribute and catalog members only. Instead, you can just reload the runtime environment to refresh the members.

**Tivoli Enterprise Monitoring Server support for audit logging**

Tivoli Management Services provides a new auditing function that you can use to capture significant events occurring in your site's IBM Tivoli Monitoring environment and record them in permanent storage for later retrieval and analysis. Each audit record fully describes some event that has changed the state of your Tivoli Monitoring system. Platforms covered include Windows, UNIX, Linux, i5/OS, and z/OS systems. On a z/OS system, the auditing facility creates and stores information in Systems Management Facility (SMF) format records. These new auditing and logging records can be stored in the Tivoli Data Warehouse. Standard reports are provided by using the Tivoli Common Reporting feature. Additionally, a new Tivoli Enterprise Portal workspace enables you to view auditing and logging records online.

For more information, see the *IBM Tivoli Monitoring: Tivoli Enterprise Portal User's Guide* and *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.*

**Web Services SOAP Server available on remote Tivoli Enterprise Monitoring Server**

The SOAP server is now started by default on all monitoring servers. This new feature is to support automation policies that send requests to the local SOAP server. However, all requests that are sent to a remote monitoring server that do not originate from a policy are rejected. For more information, see *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.*
Authorization of z/OS Take Action commands by NetView

This subject is now fully documented and unified in a single publication. For more information, see the *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS*. 
Chapter 2. Prerequisites and packaging

The information in this section helps you prepare to install and configure the OMEGAMON XE products and the components of Tivoli Management Services on z/OS systems:

- "Software and hardware prerequisites" directs you to the documentation of requirements that must be completed before you begin installing and configuring the products and components.
- "Packaging" describes the contents of the packages for the Tivoli Management Services and OMEGAMON XE products.

Software and hardware prerequisites

The Preinstallation Requirements and Instructions Technote contains a checklist of requirements and instructions for preinstallation tasks common to OMEGAMON XE products and Tivoli Management Services components being installed on z/OS systems. You can find this Technote by selecting one of your OMEGAMON XE products on the Software Support website at http://www.ibm.com/software/support and searching for "preinstallation requirements".

The Program Directory for each OMEGAMON XE product contains a complete list of the software and hardware prerequisites for that product. The Program Directory for IBM Tivoli Management Services on z/OS provides instructions for the SMP/E installation of these components and information about their z/OS hardware and software prerequisites. The hardware and software prerequisites for the distributed components of Tivoli Management Services can be found in IBM Tivoli Monitoring: Installation and Setup Guide.

To make sure that you have the latest version of all components, check for any fix packs that might be available. See the Recommended Maintenance Service Levels for OMEGAMON XE products on ITM V6.x Technote. You can find this Technote by selecting one of your OMEGAMON XE products on the Software Support website at http://www.ibm.com/software/support and searching for recommended maintenance.

Packaging

If you are installing OMEGAMON XE products and Tivoli Management Services components for the first time, you will find familiar IBM fulfillment methods (such as Shop zSeries®), installation tools (such as SMP/E or InstallShield), and installation documentation, including a program directory.

Product tapes (or their electronic representation) are in the standard IBM relfile format that IBM software manufacturing uses to create the tape images for installation systems such as ServerPac and CBPDO. If you receive your product through CBPDO, maintenance is included on the media for you to install. If you receive your product as part of a ServerPac or SystemPac®, maintenance is preinstalled.

You can order a ServerPac that includes your mainframe Tivoli products with or without the z/OS operating system. For a list of the Tivoli products you can include in a ServerPac, see the Shopz product catalog.

Important

The shared components of the IBM Tivoli Management Services on z/OS product are not automatically included in the product package for a monitoring agent on z/OS systems. Be sure to include IBM Tivoli Management Services on z/OS in your order. If you have not ordered any monitoring agents on a z/OS system but you want to install a monitoring server on a z/OS system, you can order IBM Tivoli Management Services on z/OS by itself. The IBM Tivoli Management Services on z/OS product includes both mainframe and distributed components of Tivoli Management Services (IBM Tivoli Monitoring).
Table 2 on page 13 shows the contents of a product package for Tivoli Management Services on z/OS.
<table>
<thead>
<tr>
<th>Media</th>
<th>Name and description</th>
<th>Target</th>
</tr>
</thead>
</table>
| Media set 1 of 2: **z/OS installation** | A downloaded electronic tape image provides the installation software for the mainframe components:  
- Tivoli Enterprise Monitoring Server on z/OS  
- TMS:Engine  
- Software for configuring the product components by either the PARMGEN method or the Configuration Tool (ICAT) method |  |
| | The z/OS media set also includes hardcopy license information. |  |
| Media set 2 of 2: **Distributed installation** | The distributed package includes DVDs and CDs:  
- IBM Tivoli Monitoring DVDs include subdirectories and installation procedures for components and tools on supported Windows, UNIX, and Linux operating systems.  
- IBM DB2 Universal Database™ CDs provide database functions to the components on supported Windows, UNIX, and Linux operating systems.  
- Language packs provide online help and presentation files in the supported national languages:  
  - IBM Tivoli Monitoring Language Packs DVD or CDs  
  - DB2 National Language Pack CDs | |

IBM Tivoli Monitoring publications are not included as part of the product package. You can find the latest publications in the IBM Tivoli Monitoring and OMEGAMON XE Information Center at [http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/](http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/). For a list of IBM Tivoli Monitoring publications, see Appendix F, “Documentation library,” on page 207.

An OMEGAMON XE monitoring agent on z/OS product package includes the following items:
- z/OS FMIDs either on tape or in electronic format.
  - Product tapes are in CBPDO or ServerPac format and contain product-specific FMIDs.
- Physical CDs or DVDs or electronic CD or DVD images that are required for some core product functions.
- Publications on CD or DVD as well as in hardcopy or electronic form.

Table 3 on page 14 summarizes the contents of the packages.
**Table 3. Product packaging, OMEGAMON XE monitoring agents**

<table>
<thead>
<tr>
<th>Media</th>
<th>Name and description</th>
<th>Target</th>
</tr>
</thead>
</table>
| **Media set 1 of 2: z/OS installation**                                   | Product tapes or electronic tape images provide the installation software for the monitoring agent and its components:  
  - OMEGAMON XE monitoring agent  
  - OMNIMON base (as appropriate)  
  - Shared probes (as appropriate)  
  - End to End Response Time collector (as appropriate)  |        |
|                        | The agent media set also includes hardcopy publications:  
  - OMEGAMON XE product-specific license information and the IBM International Program License Agreement (IPLA)  
  - IBM Tivoli OMEGAMON XE Monitoring Agents on z/OS: Quick Start Guide |        |
| **Media set 2 of 2: Distributed installation**                            | The agent media set includes the following DVDs or CDs or electronic DVD or CD images:  
  - OMEGAMON XE application support CD or DVD.  
  - OMEGAMON XE language support CD or DVD. |        |

Application support (which contains the predefined workspaces and situations, online help, expert advice, and OMEGAMON XE data for the Tivoli Enterprise Portal) is supplied on multiple DVDs or CDs. For a complete list, see the IBM Tivoli OMEGAMON XE Monitoring Agents on z/OS: Quick Start Guide.  
**Application support information, including the latest media levels available, can be found online at:**  
Chapter 3. Planning your deployment

The information in this chapter is intended to help you plan a deployment of the OMEGAMON XE monitoring agents and the components of Tivoli Management Services on z/OS.

Review the prerequisites in Chapter 2, “Prerequisites and packaging,” on page 11 and in the Preinstallation Requirements and Instructions Technote, and make sure they have been fulfilled. Then make the planning decisions described in this chapter:

- “Decision 1: Whether to install into a shared CSI” on page 13
- “Decision 2: Where to install your hub and remote monitoring servers” on page 16
- “Decision 3: Where to configure your monitoring agents” on page 22
- “Decision 4: Which user interface to use” on page 23
- “Decision 5: Which configuration method to use” on page 25
- “Decision 6: How to set up communications between components” on page 27
- “Decision 7: What types of runtime environments to set up” on page 35
- “Decision 8: What historical data to collect and how to manage it” on page 45
- “Decision 9: Which security options to enable” on page 48

Other sources of information

- If you are upgrading your monitoring environment, see the IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Upgrade Guide in the IBM Tivoli Monitoring and OMEGAMON XE Information Center at [http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/](http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/). In the product list on the left side of the information center, double-click Shared OMEGAMON XE publications to find the IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Upgrade Guide and other shared publications.
- For information on installing, configuring, and upgrading the distributed components of Tivoli Management Services, see the IBM Tivoli Monitoring: Installation and Setup Guide in the IBM Tivoli Monitoring section of the IBM Tivoli Monitoring and OMEGAMON XE Information Center.
- For product-specific installation, configuration, and upgrade information, see the documentation for each product in the information center and see the Technotes for each product. See Other sources of documentation on page 210 for information on accessing Technotes.

Decision 1: Whether to install into a shared CSI

To manage a suite of monitoring agent products with a monitoring server on z/OS systems, install the products into a shared consolidated software inventory (CSI) whenever possible. A shared CSI has two main advantages:

- A shared CSI eliminates duplication and can reduce space requirements by as much as 75%. The monitoring agents share components that must be duplicated if different target and distribution zones are used.
- In a shared CSI, SMP/E can automatically manage IFREQ situations across product components. Otherwise, the IFREQ situations must be managed by running SMP/E cross-zone reports. For information about cross-zone reports, see the IBM SMP/E for z/OS documentation.

Note: While most of the monitoring agents on a z/OS system can coexist in a shared CSI, IBM cannot guarantee that these products can coexist with products from other vendors.
If you install a product or component into an existing CSI that contains a previous version of the same product or component, SMP/E deletes the previous version during the installation process. To maintain more than one version of products or components, you must install them into a separate CSI from the CSI that contains the previous version.

If you decide to install into a shared CSI, follow these guidelines:

- Specify the same high-level qualifier for the target and distribution libraries.
- Make sure you have enough DASD. If you are installing into an existing shared CSI, pay particular attention to the DKAN* and TKAN* common libraries. The DKAN* and TKAN* libraries were originally allocated with secondary space allocation, to allow them to expand when you install additional products or apply maintenance. However, if these libraries have already gone into multiple extents, you might not have sufficient extents available for any necessary expansion. In that case, you might receive error messages during installation.
- During normal SMP/E processing, VSAM control interval and control area splits can occur. The resulting fragmentation can degrade SMP/E performance and space utilization. To reorganize the CSI, use your site's approved utility and method for managing VSAM files.

If you are not installing into a shared CSI, you must use a different high-level qualifier for the target and distribution libraries.

**Decision 2: Where to install your hub and remote monitoring servers**

The hub monitoring server is the focal point for the entire monitoring environment. This server is under a significant load. The hub is responsible for includes coordinating communication with remote monitoring servers, with the IBM Tivoli OMEGAMON enhanced 3270 user interface, with the Tivoli Enterprise Portal Server, and with local monitoring agents; authenticating users; consolidating and distributing data; storing and tracking situations and policies; and initiating and tracking all generated Take Action commands.

Place the hub monitoring server inside the data center on a high-performance network. Connectivity between the hub monitoring server and other directly connected components such as the remote monitoring servers must be fast and reliable.

Remote monitoring servers communicate only with the monitoring agents that report to them and with the hub monitoring server to which they report. Note that a remote monitoring server is remote with respect to the hub monitoring server, not necessarily with respect to the monitoring agents. If monitoring agents are installed on the same system as a remote monitoring server, that monitoring server is local to the monitoring agents but remote to the hub.

The load on remote monitoring servers is typically low. Load is driven higher if historical data collection is performed on the monitoring servers instead of on the monitoring agents.

You can install monitoring servers on z/OS, Windows, and some UNIX and Linux systems. See *IBM Tivoli Monitoring: Installation and Setup Guide* for a complete list of supported platforms.

**Important:** Do not switch the type (hub or remote) of an already-configured monitoring server.

**Deciding where to place the hub monitoring server**

Many organizations prefer the reliability and availability characteristics of the z/OS platform for the hub monitoring server. If most of your monitoring agents are on a z/OS system, placing the hub monitoring server on a z/OS system can shorten the communications path.

Alternatively, if most of your monitoring agents are on distributed systems, you might prefer a distributed platform for your hub monitoring server. If you install the hub monitoring server on a distributed system, such as a Windows, Linux, or AIX system, you have the option of deploying the portal server on the same system to shorten that communications path.
### Best practices

Configure your components so that only local monitoring agents and remote monitoring servers report to the hub monitoring server. Configure monitoring agents to report to a hub only if that hub is on the same system as the monitoring agents (that is, if the monitoring agents are local to the hub). If your hub is not z/OS system, then it may make sense to have a remote monitoring server on your z/OS LPARs.

### Self-describing agents

The Tivoli Enterprise Monitoring Server now supports self-describing agents, which include all the necessary application support files for the agents. After startup, self-describing agents automatically push application support updates to the self-describing hub monitoring server, the Tivoli Enterprise Portal Server, and the Tivoli Enterprise Portal browser client. Self-describing monitoring agents eliminate the requirement to recycle the monitoring server after application support updates, and ensure that application support files are current. By default, the self-describing agent feature is enabled within any remote monitoring server and any monitoring agent that provides self-describing support but the feature is disabled at the hub monitoring server. If your enterprise includes self-describing agents, you can enable the self-describing agent feature by changing parameter values for the hub monitoring server.

If you are planning to configure a hub monitoring server on a z/OS system and enable the self-describing agent feature, you must create Hierarchical File System/zSeries File System (HFS/zFS) directories on z/OS UNIX System Services (USS) systems. In this way, the hub monitoring server can process the self-describing application support updates. The USS system where you create these directories must have access to a Java runtime environment running under IBM's 31-bit or 64-bit Java SDK Version 5 (or higher) on an HFS or zFS file system. Ensure you complete the following steps before starting the configuration of the hub runtime environment:

1. Confirm that you have enough USS directory space. The amount required depends on the amount of agents you have deployed in your environment. For more information, see *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS*.
2. Install a supported level of Java and note the USS path for your Java home directory.
3. Determine the data set name of your SBXEXEC library.
4. Determine the USS path prefix where your runtime environment will store self-describing agent packages. These packages do not need to be created in advance because the configuration process will create them if needed.

You must use the `tacmd` command-line interface to administer the self-describing agent process, for example, to check the status and version of application support installed. This component is automatically installed when you install the Tivoli Enterprise Portal desktop client, the Tivoli Enterprise Portal Server, or the Tivoli Enterprise Monitoring Server after invoking the GUI installer on distributed operating systems. If you are not installing any of these components into your monitoring environment, you must install the Tivoli Enterprise Services user interface extensions component of the Tivoli Enterprise Management Agent feature on a distributed system because it is not available on z/OS systems. For more information, see z/OS in the *IBM Tivoli Monitoring: Installation and Setup Guide*.

The HTTP or HTTPS port must be enabled at the monitoring server to allow the `tacmd` command-line utility to connect through the SOAP server interface. To enable the SOAP server, include the following steps in the following table during the configuration of your monitoring server runtime environment depending on your choice of configuration method:
Table 4. Configuration methods

<table>
<thead>
<tr>
<th>Configuration method</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARMGEN</td>
<td>Enable the KDS_TEMSOAP_SERVER_FLAG parameter.</td>
</tr>
</tbody>
</table>
| Configuration Tool (ICAT) interactive mode | 1. From the Configure the TEMS menu, navigate to the Specify configuration values option.  
2. From the next panel, select F5=Advanced.  
3. On the next screen, in the Enable Web Services SOAP Server field, specify Y. |
| Configuration Tool (ICAT) batch mode   | Enable the KDS_CMS_CTSOAP parameter.                                  |

If you are planning to configure a hub monitoring server on a z/OS system, enable the self-describing agent feature and use the IBM Tivoli OMEGAMON enhanced 3270 user interface. You are not required to install a Tivoli Enterprise Portal Server for the sole purpose of adding application support files. To administer the self-describing agent feature, you must install the Tivoli Enterprise Services user interface extensions component into a distributed machine. This component provides the tacmd command-line utility to interact with a monitoring server. This component is available on the IBM Tivoli Monitoring DVD image.

For more information, see the *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS* and the *IBM Tivoli Monitoring: Command Reference*.

**High-availability hub monitoring server**

An operational hub monitoring server is essential to a monitoring environment. If the hub monitoring server address space fails, or if the system on which the hub is installed has a planned or unplanned outage, the flow of monitoring data comes to a halt. Therefore, it is important to restart the hub or move it to another system as quickly as possible.

You can configure a high-availability hub monitoring server in any sysplex environment with dynamic virtual IP addressing (DVIPA) and shared DASD. A high-availability hub is configured in its own runtime environment, without any monitoring agents, and can be configured on the same LPAR with a remote monitoring server. This configuration allows the hub monitoring server to be relocated to any suitable LPAR in the sysplex with no changes, and with minimal disruption to the components connecting to the hub.

**Best practices**

Do not enable system variables for a high-availability hub monitoring server. The recommended best practice is to create a high-availability hub runtime environment if all of the following conditions apply:

- You have installed and intend to run OMEGAMON XE on z/OS and
- You intend to use the self-describing agent feature for application support installation and
- You plan to run a hub monitoring server on z/OS (as opposed to a distributed platform)

Additionally, configure a remote monitoring server running an OMEGAMON on z/OS agent on all LPARs that you want to monitor. If you elect not to create a high-availability hub runtime environment, a recycle may be required when you install maintenance updates to applications, such as OMEGAMON on z/OS, on the hub monitoring server. A recycle might cause a temporary downtime within your monitoring environment.

Figure 2 on page 19 shows a typical configuration with a high-availability hub runtime environment deployed.
In this monitoring environment, you have two LPARs: LPAR 1 and LPAR 2. In LPAR 1, a runtime environment (RTE A) has been created containing just the hub monitoring server, which has been defined as a high-availability hub monitoring server with DVIPA. Because you also want to monitor LPAR A in addition to the subsystems running on it (in this example, CICS), a second runtime environment (RTE B) is created with a remote monitoring server and any monitoring agents that are needed.

Note: The OMEGAMON on z/OS monitoring agent shares an address space with the remote monitoring server while the OMEGAMON for CICS monitoring agent runs within its own address space.

On the second LPAR, a runtime environment (RTE C) is created in the same fashion as RTE B to monitor the systems and subsystems on that LPAR. It connects to the high-availability hub monitoring server through the DVIPA address. The advantage of the high-availability configuration is that if anything happens to LPAR 1, either planned or unplanned, the hub can be restarted on LPAR 2 without the need for reconfiguring the existing runtime environments.

For detailed information about planning for and configuring a high-availability hub on z/OS systems, see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS. For detailed information about the high-availability hub on distributed systems, see IBM Tivoli Monitoring: High-Availability Guide for Distributed Systems.

Deciding where to place remote monitoring servers
It is a good idea to place a remote monitoring server on every z/OS system where you are installing monitoring agents. In fact, two monitoring agents (OMEGAMON XE on z/OS and OMEGAMON XE for Storage on z/OS) require that you configure them in the same address space as a hub or remote monitoring server.

For advice about placing remote monitoring servers on distributed systems, see the IBM Tivoli Monitoring: Installation and Setup Guide.
Tips

- A remote monitoring server can report to a high-availability hub on the same LPAR.
- If the hub is not a high-availability hub, it cannot be on the same LPAR as any of the remote monitoring servers that report to it and that are configured to use any TCP/IP protocol.
- If more than one remote monitoring server is configured in a z/OS image and if a TCP/IP protocol is being used for communication, the hub to which each remote monitoring server reports must have a unique port number. Otherwise, connectivity problems are likely.
- If more than one hub is configured in a z/OS image, each hub must have a unique port number for any nonsecure TCP/IP protocols being used and a unique port number for any secure TCP/IP protocols being used.

For information about port number allocation, see "Port number assignments" on page 31.
- If a remote monitoring server is to communicate with any monitoring agents that require SNA, the remote monitoring server must be configured for SNA communications. Examples of such monitoring agents include OMEGAMON XE on z/OS (for the EPILOG facility of the OMEGAMON II component) and OMEGAMON XE for Messaging on z/OS (for the 3270 interface component). See the product-specific configuration guides for further information about SNA requirements.

Monitoring server names

Each monitoring server has a unique name used for internal processing. This name is known as the TEMS name or the CMS name. You specify the TEMS name for a monitoring server on a z/OS system when you define the runtime environment in which the monitoring server is to be configured. The default name is &rte:CMS (where &rte is the name of the runtime environment). The TEMS name is stored because the value of the CMS_NODEID environment variable in the KDSENV member of the &rhilev.&rte.RKANPARU data set.

In general, it is best to accept the default TEMS name. If you want to specify a different name, follow these guidelines:
- The TEMS name must be unique.
- The name is alphanumeric and must begin with an alphabetic character.
- The length of the name must be at least 2 characters and no more than 32 characters.
- The name cannot contain blanks or special characters ($#@). An underscore (_) is permitted and conforms to ISO 9660 standards. A period (.) is also valid.
- The TEMS name is case-sensitive on all platforms. If you use a mixed-case name, you must supply the same mixed-case name when you configure all components and monitoring agents that will connect to the monitoring server.

Planning for the SOAP server on z/OS systems

The Simple Object Access Protocol (SOAP) is a communication method that uses the Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML) as the mechanisms for information exchange. Because web protocols are installed and available for use by all major operating system platforms, HTTP and XML provide a ready solution to the problem of how programs running under different operating systems in a network can communicate with each other. SOAP specifies exactly how to encode an HTTP header and an XML file so that a program in one computer can call a program in another computer and pass it information. It also specifies how the called program can return a response.

An advantage of SOAP is that program calls are likely to get through firewall servers that screen out requests other than those for known applications (through the designated port mechanism). Because HTTP requests are usually allowed through firewalls, programs using SOAP to communicate can generally communicate with programs anywhere.
The SOAP server is an application server plug-in that receives and sends XML data, and provides XML SOAP interfaces into the Tivoli Management Services components and the monitoring agents. The SOAP server is installed with each monitoring server and is enabled during configuration of the hub monitoring server. During configuration, you specify the list of non-local hub monitoring servers with which the local SOAP server can communicate.

**SOAP server terminology**
To configure the SOAP server, you must understand the following terms:

**Hub monitoring server list**
Aliasing mechanism for identifying which non-local hub monitoring servers can be accessed from the local SOAP server.

**KDSTHUBS**
Global table used by all SOAP servers enabled in the installation library.

**KSHXHUBS**
Member stored in the &rhilev.&rte.RKANPARU library and containing the hub monitoring server list.

**User access list**
List of user IDs associated with the hub monitoring server list.

**SOAP server configuration and security**
SOAP server configuration creates a KSHXHUBS member in the &rhilev.&rte.RKANPARU library. The KSHXHUBS member contains the hub monitoring server list, an aliasing mechanism for identifying the hub monitoring servers with which the local SOAP server can communicate.

User access to a SOAP server can be secured in one of two ways: by enabling security and creating user accounts for the hub monitoring server, or by adding specific users to the SOAP server definition. If security is not enabled and no users are added to the server definition, the SOAP server honors all requests from any sender. If security is enabled on the hub monitoring server, the SOAP server honors requests only from users defined to the system authorization facility. However, if any users are added to the SOAP server definition, only those users have access to the server, regardless of whether security is enabled on the monitoring server.

When enabling access to a non-local hub monitoring server from the SOAP server, you can choose one of these options:
- Enable global access to all user IDs that pass logon validation.
- Specify a hub monitoring server list and, for each monitoring server on the list, the user IDs that are allowed to query (read) or update (write to) that monitoring server.

You must still use external security to validate user IDs and passwords, after the user IDs pass validation with the KSHXHUBS member.

The hub monitoring server list in a runtime environment is maintained in the KDSTHUBS global table. KDSTHUBS is used by all SOAP servers that are enabled in the installation library. Any changes you make to the hub monitoring server entries in KDSTHUBS affect KSHXHUBS members used in the different RKANPARU libraries for the runtime environments, when you reconfigure those runtime environments.

**Important:** Do not edit the KSHXHUBS member directly. Its XML tags and values require a specific format and are case-sensitive. If you want to change the contents of the KSHXHUBS member, do so in the PARMGEN configuration profile or in the Configuration Tool.
**Specifying communication protocols for the SOAP server**

If you enable the SOAP server while configuring the hub, at least one of the communication protocols must be either IP*.PIPE or IP*.UDP. The communication protocols for the SOAP server are automatically initialized to the protocol values set for the runtime environment.

**Decision 3: Where to configure your monitoring agents**

Two monitoring agents (OMEGAMON XE on z/OS and OMEGAMON XE for Storage on z/OS) require that you configure them in the same address space as a hub or remote monitoring server. For the other monitoring agents, you have the option to configure each monitoring agent as *stand-alone* (in its own address space) or in the same address space with a monitoring server.

**Terminology tip:** The term *stand-alone* can be confusing, because it has one meaning when applied to a monitoring server and another meaning when applied to a monitoring agent. A *stand-alone monitoring server* is one configured in its own runtime environment, without any monitoring agents. A *stand-alone monitoring agent* is one configured in its own address space, rather than in the same address space with a monitoring server. A stand-alone monitoring agent can be in the same runtime environment with a monitoring server, but a stand-alone monitoring server is never in the same runtime environment with monitoring agents.

**Best practices**

- Unless you are planning to configure a high-availability hub, configure one runtime environment per logical partition (LPAR). In each runtime environment, configure a monitoring server (hub or remote) and all agents required for monitoring the various workloads on that system. Configure all monitoring agents to report to the monitoring server in their runtime environment. See *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS*.

- Unless a monitoring agent is required to run in the same address space as a monitoring server, it is best to configure each monitoring agent stand-alone (in its own address space). Configuring the monitoring agents stand-alone has several advantages:
  - A stand-alone monitoring agent can be started and stopped independent of the monitoring server.
  - If a monitoring server fails or becomes unresponsive, stand-alone agents that report to it can switch to a secondary monitoring server specified during configuration.
  - You can apply maintenance to a stand-alone monitoring agent without interfering with the operation of other components.
  - Troubleshooting is easier if each monitoring agent is identified in trace logs by its own started task.

Even if you configure a monitoring agent stand-alone, it still might report to the same monitoring server as other monitoring agents. Therefore, make sure that the values you supply when configuring the monitoring server are compatible with the requirements of all the monitoring agents intended to report to that monitoring server. For example, some monitoring agents may require a SNA connection between the agent and the monitoring server.
Self-describing agents
The Tivoli Enterprise Monitoring Server now supports self-describing agents, which include all the necessary application support files for the agents. After startup, self-describing agents automatically push application support updates to the self-describing hub monitoring server, the Tivoli Enterprise Portal Server, and the Tivoli Enterprise Portal browser client.

Self-describing monitoring agents eliminate the requirement to recycle the monitoring server after application support updates, and ensure that application support files are current. By default, the self-describing agent feature is enabled within any remote monitoring server and any monitoring agent that provides self-describing support but the feature is disabled at the hub monitoring server. If your enterprise includes self-describing agents, you can enable the self-describing agent feature by changing parameter values for the hub monitoring server.

Note: Not all OMEGAMON XE agents currently provide self-describing agent support. See the product-specific Planning and Configuration guides for confirmation of support.

For more information, see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.

Autonomous agents
You can choose to configure stand-alone monitoring agents to run in autonomous mode (without communicating directly with a monitoring server). An autonomous agent can emit Simple Network Management Protocol (SNMP) traps and Event Integration Facility (EIF) events directly to a Netcool/OMNIbus ObjectServer for agent-specific situations (but not for enterprise situations). Autonomous agents can be advantageous in environments where disk space or transmission bandwidth is in short supply.

An autonomous agent requires the DSEVT DDNAME in the monitoring agent started task in the &rhilev.&rte.RKANSAMU data set. This DDNAME points to the &rhilev.&rte.RKDSEVT data set.

The IBM Tivoli Monitoring: Installation and Setup Guide provides instructions for configuring Netcool/OMNIbus ObjectServers to receive the events. For information on specifying which situation events to forward, see the Tivoli Enterprise Portal online help and the IBM Tivoli Monitoring: Tivoli Enterprise Portal User's Guide.

For information about parameters to set for an autonomous agent, see the "Agent autonomy" chapter of the IBM Tivoli Monitoring: Administrator's Guide. You can add these parameters for an autonomous agent on a z/OS system by editing the KppSPENV member of the PARMGEN WCONFIG library or by using the Specify Nonstandard Parameters panel in the Configuration Tool (ICAT) (see "Adding, changing, or deleting a parameter in a runtime member" on page 98).

For instructions on enabling SNMP V3 passwords for autonomous agents, see "Enabling SNMP V3 passwords for autonomous agents" on page 112.

Decision 4: Which user interface to use
Most OMEGAMON XE monitoring products support 3270 "green screen" interfaces as well the graphical Tivoli Enterprise Portal interface, which is based on distributed platforms. The two types of interfaces differ in the information they offer, the infrastructure they require, and the security they provide. The capabilities of each interface, as well as the familiarity of a user community with a specific type of interface, should be considered when deciding which interface to provide.

For many products, it is possible to provide several types, allowing users to access the product in the manner more appropriate to their skills and familiarity. The Tivoli Enterprise Portal interface requires that you that you install one or more Tivoli Enterprise Monitoring Servers and a Tivoli Enterprise Portal Server,
which must be installed on a distributed platform. The IBM Tivoli OMEGAMON enhanced 3270 user interface requires that you install one or more Tivoli Enterprise Portal Servers, but it offers plex-level as well as system-level and cross-product data and enhanced functionality. The older 3270 user interfaces require less infrastructure, but these interfaces provide more limited information and customization options. Review the following sections to decide which type of interface to use.

**IBM Tivoli OMEGAMON enhanced 3270 user interface**

The OMEGAMON enhanced 3270 user interface retains the virtues of the earlier 3270 interfaces, while offering many of the features of the Tivoli Enterprise Portal in a native 3270 interface:

- The ability to view plex-wide data, data from multiple systems, and data from multiple agents in the same interface or even the same workspace
- The ability to modify product-provided workspaces to create customized views and queries, and or create new workspaces
- The ability to set dynamic filters
- The ability to define or modify thresholds that trigger status indicators

The OMEGAMON enhanced 3270 user interface requires that at least one hub monitoring server must be configured, and for OMEGAMON XE on z/OS, you must install a monitoring server in each LPAR.

**OMEGAMON (3270) and OMEGAMON II (CUA) interfaces**

The older 3270 interfaces require the least amount of product-specific architecture to support. These interfaces perform well and are user-friendly to systems programmers and administrators that are used to an ISPF 3270 style interface. Standard and custom PF Key settings, menu options, and command-line interface options allow for short cuts to commonly viewed screens. While basic customization options allow for highlights and other eye-catcher techniques to be added to the interface, the customization options are limited. The 3270 interfaces use dedicated connection features and avoid disk I/O. Documentation may be out-of-date or difficult to find.

The OMEGAMON (3270) and OMEGAMON II interfaces do not provide plex data, cross-LPAR monitoring, or cross-product views. However, you can get limited multisystem monitoring and cross-system views using OMEGAMON DE. The following OMEGAMON XE products provide older 3270 interfaces:

- IBM Tivoli OMEGAMON XE on z/OS
- IBM Tivoli OMEGAMON XE for CICS on z/OS
- IBM Tivoli OMEGAMON XE for DB2 on z/OS
- IBM Tivoli OMEGAMON XE for IMS on z/OS
- IBM Tivoli OMEGAMON XE for Messaging on z/OS
- IBM Tivoli OMEGAMON XE for Storage on z/OS
- IBM Tivoli OMEGAMON XE for Mainframe Networks on z/OS

OMEGAMON XE on z/OS and OMEGAMON XE for Storage require more infrastructure than the other agents: a Tivoli Enterprise Monitoring Server must be installed in the same runtime environment as each monitoring agent and the OMEGAMON XE component to be configured. (While the OMEGAMON XE on z/OS agent can be configured to run as a stand-alone system, that is, outside of a monitoring server, such a configuration does not support plex data.)

The other agents do not require either a portal server or a monitoring server, and require only configuration of the OMEGAMON II component.

**Tivoli Enterprise Portal interface**

This graphical user interface may be accessed through a Java-based desktop client or a supported browser. The Tivoli Enterprise Portal provides the following features, including:
• The ability to merge the data from multiple agents or multiple systems into a single workspace for business views and unit of work views
• The ability to define specific conditions that should be monitored (situations)
• The ability to define colored highlights and sounds to alert operators when events occur
• Proactive automation to respond to situations
• Highly customizable graph and table views
• Integral historical data collection, reporting, and warehousing
• Customizable historical reporting
• Integration with other Tivoli and IBM products, including single sign-on
• Event forwarding to event management systems LDAP authentication

The portal uses the standard browser interface and is user-friendly to most users and is often the preferred interface for operation centers, managers, and technical staff. Using the Tivoli Enterprise Portal interface always requires defining and running the Tivoli Enterprise Portal Server, a hub Tivoli Enterprise Monitoring Server, and potentially one or more remote monitoring servers. The agents that will provide data also have to be configured. For more information about the Tivoli Enterprise Portal, see the IBM Tivoli Monitoring: Tivoli Enterprise Portal User’s Guide.

Decision 5: Which configuration method to use

To configure the Tivoli Enterprise Monitoring Server on a z/OS system, the IBM Tivoli OMEGAMON enhanced 3270 user interface and any OMEGAMON XE monitoring agents with a runtime environment, you have the choice between two configuration methods. Both provide the ability to vary the parameters and settings specific to your runtime environment and generate the started tasks and libraries required to run your monitoring environment. The following configuration methods are available:

• **PARMGEN Workflow user interface:** This method is now the preferred method of product configuration. Using the PARMGEN Workflow user interface, you edit a comprehensive list of parameters for configuring all installed products and components. You then submit a series of jobs to create a complete runtime environment with the parameter values you specified.

  The PARMGEN method can be used for creating new runtime environments and for upgrading existing ones that were created from product versions for which the PARMGEN configuration method was enabled. To upgrade from a product version lower than the first PARMGEN-enabled version, you must use the Configuration Tool (ICAT). To determine the lowest PARMGEN-enabled version of each of your products, see the PARMGEN Technote at [http://www.ibm.com/support/docview.wss?uid=swg21417935](http://www.ibm.com/support/docview.wss?uid=swg21417935).

  For information about configuring your runtime environment using the PARMGEN Workflow user interface, see Chapter 4, “Configuring products using the PARMGEN Workflow user interface,” on page 55.

• **Configuration Tool (ICAT) method:** The Configuration Tool (ICAT) is included with several releases of Tivoli Management Services and OMEGAMON XE. The tool provides a batch mode and an interactive mode to configure a runtime environment.

  With batch mode, you can create a single batch job to build, configure, and load a runtime environment for the Tivoli Enterprise Monitoring Server, monitoring agents, or both. See the information in Part 3, “Configuration Tool (ICAT) scenarios,” on page 119, for more information about batch mode configuration.

  In interactive mode, ISPF panels guide you through the steps of setting the parameter values and of tailoring and running the configuration jobs. For more information about configuring your runtime environment using the Configuration Tool (ICAT), see the information in Chapter 5, “Configuring products with the Configuration Tool (ICAT),” on page 77.

Either the PARMGEN method or the Configuration Tool batch mode is useful for replicating a configured runtime environment to other z/OS systems. The PARMGEN method is preferable for configuring all the products in a runtime environment and then replicating the configuration to other z/OS systems.
Best practices
For the OMEGAMON XE version 4.2.0 and subsequent releases, both the PARMGEN method and the Configuration Tool (ICAT) method are supported.

If you have runtime environments that are already configured by the Configuration Tool (ICAT) method, a conversion tool (the KCIJPCNV job) is provided for using the existing parameter values in those runtime environments to set up initial values for new runtime environments to be configured by the PARMGEN method. After you use the PARMGEN method to configure runtime environments, you cannot use the Configuration Tool (ICAT) to edit or maintain them.

Configuration parameters
Whether you decide to use the PARMGEN or Configuration Tool (ICAT) configuration method, the work of configuring your products consists primarily of setting appropriate values for the configuration parameters of runtime environments and of the product components they contain. This section provides a basic explanation of the configuration parameters you will use in performing the tasks described in the following chapters.

For reference information about the parameters used in configuring runtime environments and Tivoli Enterprise Monitoring Servers, see IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Common Parameter Reference. For reference information about the parameters used in configuring monitoring agents, see the documentation for each product.

Where configured parameters are stored
Most configuration parameters and their configured values are stored in the KppENV or KppSYSIN members of the &rhilev.&rte.RKANPARU data set for each runtime environment, where:

- pp is the 2-character product or component code. For a comprehensive list of the product codes, see Appendix D, “Product codes,” on page 201.
- &rhilev is the high-level qualifier for the runtime environment data sets
- &rte is the runtime environment name, which is used as the mid-level qualifier for the runtime environment data sets

For information about runtime environments, see Decision 7: What types of runtime environments to set up on page 35.

The parameters stored in the KppENV member are environment variables, which determine the operating characteristics of the runtime environment in which products and components are configured. The parameters stored in the KppSYSIN member are startup parameters, which determine the default startup values for each product or component.

Some environment variables and startup parameters are stored in members other than KppENV and KppSYSIN, or in data sets other than RKANPARU. Additionally, some parameters that are neither environment variables nor startup parameters must be included in the runtime libraries for the products and components.

System variables
System variables are elements that allow systems to share parameter definitions while retaining unique values in those definitions. System variables act like variables in a program; they can take on different values, based on the input to the program. When you specify a system variable in a shared parameter definition, the system variable acts as a placeholder. Each system that shares the definition replaces the system variable with a unique value during initialization.
If you use system variables, the components inherit the system values of the system on which they are started (the host z/OS system). These system-specific values are then automatically loaded into temporary data sets that exist only while the component runs. The result is that the software runs correctly by using the system-specific parameter values for the host z/OS system.

Using system variable support includes the following benefits:
- You can deploy the same software unit, consisting of any or all TMS:Engine-based products, on any system without modification. LPAR-specific values are automatically resolved and substituted at product startup.
- The number of unique runtime environments required is smaller (although unique physical data sets must still exist.) This feature saves storage space, CPU, and labor.
- The same started task JCL and the same VTAM® node can be used on any system without modification.
- You can choose to use a single VTAM major node in place of the individual product major nodes. When generated, a single VTAM major node contains all VTAM applids for all TMS:Engine-based products you have configured in the runtime environment.

**Tips**
- You cannot use system variables in the runtime environment of a high-availability hub. (Because the high-availability hub can be started on any LPAR with DVIPA, parameters such as the monitoring server node ID, name, and VTAM major node must be set to static values. See *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS* for details.)
- Product started tasks contain a preprocessing step that resolves all system variable specifications in the product parameter members.
- If you enable system variables, product parameter members contain many variables whose values are resolved during startup of the started task.

See the *z/OS MVS™ Initialization and Tuning Reference* (SA22-7592) for basic information on system variables.

Both the PARMGEN configuration method and the Configuration Tool (ICAT) method support system variables. See Appendix A, “Enabling system variable support,” on page 165.

**Decision 6: How to set up communications between components**

A number of communication protocols are available for connections between the hub and remote monitoring servers and between each monitoring agent and the monitoring server it reports to. You configure each monitoring server to use one or more of these protocols, and you assign a priority number to each protocol selected. When the monitoring server attempts to communicate with another component, it tries the highest-priority protocol first. In case of failure, it goes on to the second-priority protocol, and so on. Every component configured to communicate with a monitoring server must have at least one communication protocol in common with that monitoring server.

The communication protocols are of two basic types:
- **SNA.** Because some monitoring agents on z/OS systems require SNA, it must be one of the protocols selected for a monitoring server communicating directly with those monitoring agents. However, SNA does not have to be the highest-priority protocol.
- **TCP/IP.** The TCP protocols available are IP.PIPE, IP.SPIPE, IP6.PIPE, IP6.SPIPE, IP.UDP, and IP6.UDP. For the hub monitoring server, at least one of the protocols chosen must be a TCP protocol to support the SOAP server. If you plan to implement long-term historical data collection, communication with the Tivoli Data Warehouse also requires a TCP protocol.
IP.PIPE is the default protocol. All of the "piped" protocols (IP.PIPE, IP.SPIPE, IP6.PIPE, and IP6.SPIPE) enable the monitoring software to traverse most firewall configurations. If you choose piped protocols for the monitoring server and monitoring agents, be aware of the following limitations:

- The maximum number of piped processes per host is 16.
- The piped protocols use only one physical port per process.

If you are configuring a high-availability hub monitoring server, Dynamic Virtual IP Addressing (DVIPA) is required on the z/OS system. This allows the high-availability hub to be moved and restarted on, for example a different LPAR, without the need for reconfiguration should problems occur or if you have a planned maintenance outage on the LPAR where it was originally running. DVIPA requires a piped TCP/IP protocol to be used and the address of the monitoring server must be resolvable through the Domain Name Server (DNS).

Your communication protocol settings for a monitoring server on a z/OS system are saved in the KDE_TRANSPORT environment variable of the KDSENV member of the &rhilev.&rte.RKANPARU library for the runtime environment.

**Requirements for TCP/IP communication protocols**
Review the following TCP-related requirements.

**Default OMVS segment**
To use the TCP/IP communication protocols, a Tivoli Enterprise Monitoring Server on a z/OS system requires a default OMVS segment. See the z/OS Communications Server IP Configuration Guide for an explanation of how to provide an OMVS segment.

**TCP/IP stack name**
You are prompted for a TCP/IP started task name when you create a runtime environment and when you configure a monitoring server or monitoring agent. In a new runtime environment, the default value for the TCP/IP started task name is an asterisk (*). This default, which uses the first TCP/IP stack that was started, is suitable if the LPAR contains a single TCP/IP stack.

If the LPAR contains more than one TCP/IP stack, you can specify the started task name of the TCP/IP stack you want to use; or you can specify the number 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.

**Best practices**
Whenever 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.

**Implementation of firewall support**
Tivoli Management Services supports most common firewall configurations, including those that use address translation (application proxy firewall is a notable exception). To enable this support, use the piped protocols, which open a single port on the firewall for communication by IBM products. If your environment includes a firewall between any components that must communicate with each other, you must specify at least one piped protocol during configuration.

During startup, the monitoring server registers its services and the IP address of these services with a location broker. Clients such as monitoring agents send queries to the location broker to request address information for a service, and receive a list of protocols and IP addresses at which these services are
available. The client then sends a specific server request to one of the addresses in the list received from the location broker. Service registration with the location broker assumes address continuity.

If the published address of the monitoring server is identical and reachable for either side of the firewall, then nothing further has to be done to achieve communications in this firewall environment. If the same address cannot be reached from either side of the barrier firewall, then either ephemeral pipe support or broker partitioning is required.

**Ephemeral pipe support**

By default, ephemeral pipe support is enabled when IP*.PIPE connections cross a network address-translating (NAT) firewall. Ephemeral pipe support allows piped connections to cross a NAT firewall without a broker partition file. If this default does not suit your network, you can disable ephemeral pipe support and use location broker partitioning instead, or you can force ephemeral connections to be used even when piped communications do not cross a NAT firewall.

*Disabling ephemeral pipe support:* Ephemeral pipe support and location broker partitioning for address translation are mutually exclusive. To disable ephemeral pipe support, you specify Y as the value of the KDS_TEMS_COMM_ADDRESS_XLAT parameter in the PARMGEN configuration profile, or in the Configuration Tool (ICAT) as the value of the Address translation field of the IP.PIPE configuration panels. See [Broker partitioning](#) for information about enabling address translation through a broker partition file.

*Forcing ephemeral connections:* Under some circumstances, you might want to force ephemeral connections to be used for IP*.PIPE communications, even if they do not cross a NAT firewall. For example, if your configuration requires more than 16 piped processes on a host, ephemeral connections make it possible to exceed the TCP maximum.

However, forcing ephemeral connections can have serious drawbacks:

- If an ephemeral connection breaks between a monitoring agent and the monitoring server to which it reports, the monitoring server might not be able to originate a connection back to the monitoring agent.
- The address of a monitoring agent configured for ephemeral connections is randomly assigned in the monitoring server logs. Tracing and troubleshooting are more difficult as a result.
- Extra configuration might be required for enabling communications between the monitoring agents and the Warehouse Proxy agent.

If you want to force ephemeral connections, use the EPHEMERAL option of the KDE_TRANSPORT environment variable.

- EPHEMERAL:Y or EPHEMERAL:OUTBOUND forces outbound connections to be ephemeral.
- **Important:** You can specify EPHEMERAL:Y or EPHEMERAL:OUTBOUND for a monitoring agent or a remote monitoring server, but not for a hub.
- EPHEMERAL:INBOUND forces inbound connections to be ephemeral.

You can specify the EPHEMERAL option in the Kpp_X_KDE_TRANSPORT_OPTIONS parameter in your PARMGEN configuration file or by using the Specify Nonstandard Parameters panel in the Configuration Tool (ICAT) (see [Chapter 4, “Configuring products using the PARMGEN Workflow user interface,” on page 55](#) and “Adding, changing, or deleting a parameter in a runtime member” on page 98).

For information about ephemeral pipe support on distributed components, see *IBM Tivoli Monitoring: Installation and Setup Guide*.

**Broker partitioning**

Address translation is an enhanced security feature of some firewall configurations. With this feature, components that must be reached across the firewall have two unique but corresponding addresses: the
external address (valid for components outside the firewall) and the internal address (valid for components inside the firewall). A component on either side of the firewall knows only about the address that is valid for its partition (its own side of the firewall).

You can configure broker partitioning during configuration of the monitoring server on a z/OS system. To do so, you specify Y as the value of the KDS_TEMS_COMM_ADDRESS_XLAT parameter in the PARMGEN configuration profile, or (in the Configuration Tool (ICAT)) as the value of the Address translation field of the IP.PIPE configuration panels. You also supply, as the value of the KDS_TEMS_PARTITION_NAME parameter in the PARMGEN configuration profile or in the Configuration Tool (ICAT) Partition name field, the label that identifies the location of the monitoring server relative to the firewalls used for address translation. The partition name that you supply is added to the partition table, which contains labels and associated socket addresses provided by the firewall administrator. The label is used outside the firewall to establish monitoring server connections.

Additionally, you supply the IP address of the monitoring server in its own partition, and the partition name and address assigned to the monitoring server from a location on the other side of each firewall being used. These values are saved as the KDC_PARTITIONFILE environment variable in the KDSENV member of the &rhilev.&rte.RKANPARU library. KDC_PARTITIONFILE points to a new member, KDCPART, created in the &rhilev.&rte.RKANPARU library.

Then, when you configure a monitoring agent that reports to the monitoring server, you specify Y as the value of the address translation parameter, and you supply the partition label of the monitoring server. These values are saved as the KDC_PARTITION environment variable in the monitoring agent's KppENV member of the &rhilev.&rte.RKANPARU library.

The well-known port for the hub monitoring server must be authorized by the firewall administrator. For the IP*.PIPE protocols, no additional ports require authorization. For the IP*.UDP protocols, a range of UDP ports must be authorized.

Firewall gateway support
A firewall gateway provides end-to-end connectivity options for environments with specific TCP/IP connection management policies. The firewall gateway can negotiate numerous firewall hops and supports network address translation. You can use a firewall gateway to configure network traffic so that it is always initiated from the more secure network zone, if two communicating components are in zones with different security levels.

A firewall gateway can be the most advantageous firewall configuration when any of the following conditions apply:
- A single TCP connection cannot span between product components. Example: communication between components requires crossing more than one firewall in an environment with a policy that does not allow a single connection to traverse more than one firewall.
- Connection requirements do not allow the default pattern of connections to the hub monitoring server. Example: agents fail to connect to a monitoring server in a zone with higher security than that of the agents; security policy allows a connection to be established from a more secure zone to a less secure zone, but not the other way around.
- Open firewall ports must be reduced to a single port or connection. The gateway can consolidate the ports into one. Example: agent failover and monitoring server assignment must be managed symbolically at the hub monitoring server end of the connection. Because gateway connections are made between matching service names, an administrator can change the failover and monitoring server assignment of agents by changing the client proxy bindings at the hub monitoring server.

To configure the firewall gateway, you must perform two tasks:
1. Create an XML document that specifies a set of zones, each of which contains at least one server (upstream) interface with one or more embedded client (downstream) interfaces. The XML document
must be stored as a member of the &rhilev.&nte.RKANPARU library, and the member name must conform to z/OS naming standards (no more than 8 characters).

Here is an example of a gateway XML document, stored as the ZOSPROXY member of the RKANPARU library:

```xml
000002 name="ZOSproxy" threads="32">
000003 <interface name="zosproxy_upstream" role="proxy">  
000004 <zone name="trusted" maxconn="512" error="ignore">  
000005 <bind ipversion="4" localport="pool2K" service="tems_pipe">  
000006 <connection remoteport="1920">127.0.0.1</connection>
000007 </bind>
000008 <interface name="zosproxy_downstream" role="listen">  
000009 <bind ipversion="4" localport="60902">  
000010 </bind>
000011 </interface>
000012 </interface>
000013 </zone>
000014 <portpool name="pool2K">20000-21023 21024-22047</portpool>
000015 </tep:gateway>
```

For reference information about the elements of the gateway XML document, see the "XML document structure" section of the Firewalls appendix to the IBM Tivoli Monitoring: Installation and Setup Guide.

2. In the KppENV member of the &rhilev.&nte.RKANPARU library, add a KDE_GATEWAY environment variable that references the XML document.

   Example:
   ```
   KDE_GATEWAY=ZOSPROXY
   ```

   For information about adding an environment variable, see "Adding, changing, or deleting a parameter in a runtime member" on page 98.

For more information about configuring firewall support, see the following publications:

- Instructions for enabling firewall support for the z/OS components: IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS and the configuration documentation for each monitoring agent.
- Instructions for enabling firewall support for the distributed components: the Firewalls appendix to the IBM Tivoli Monitoring: Installation and Setup Guide.

**Port number assignments**

Tivoli Management Services reserves a default well-known port for the first process started on a system (normally, the monitoring server). For monitoring agents and other components, an algorithm based on the port number assigned to the monitoring server calculates the listening port to reserve. A port number cannot be assigned to a component if it is already reserved for another component or application.

The default listening port number for the monitoring server is 1918 for IP.PIPE, IP.UDP, IP6.PIPE, and IP6.UDP. For the secure IP protocols (IP.SPIPE and IP6.SPIPE), the default port number is 3660. For SNA, the default is 135. It is generally best to accept the default setting. However, you might find it necessary to change the setting under some conditions; for example, when the port assigned to a component by the algorithm is already reserved by another application or component.

**Algorithm for allocating port numbers**

Tivoli Management Services uses the following algorithm to allocate port numbers for monitoring agents and other components to use in communications with the local monitoring server, under any of the TCP protocols:
allocated port = well-known port + (N*4096)

where:

- well-known port is the port number assigned to the monitoring server (for example, the default, 1918).
- N indicates the position of the monitoring agent or other component in the startup sequence.

For example, if there are a monitoring server and two monitoring agents on a system, and the monitoring server uses port 1918, the first monitoring agent in the startup sequence is assigned port 6014 (1918 + 1*4096), and the second monitoring agent to start is assigned port 10110 (1918 + 2*4096).

**Controlling port number assignments**

If you have to change port number assignments, you can do so in either of two ways:

- Changing a TCP port number assigned to the monitoring server.
- Using the SKIP and COUNT options on the KDE_TRANSPORT environment variable.

You can also use the POOL option on the KDE_TRANSPORT environment variable to limit connections to a specific range of port numbers.

### Tips

1. If you decide to modify the KDE_TRANSPORT environment variable, it is best to do so under the guidance of IBM Software Support. See Appendix G, “Support information,” on page 211.

2. For instructions on changing parameters so that they are not overridden when products are reinstalled or reconfigured, see “Adding, changing, or deleting a parameter in a runtime member” on page 98.

**Changing a TCP port number assigned to the monitoring server:** If you change a TCP port number assigned to a monitoring server on a z/OS system, the port numbers allocated for the local monitoring agents also change, because the algorithm starts from the monitoring server's well-known port. You can use this method to avoid conflicts with ports reserved for other applications or components. If you specify a high number, conflicts are less likely than if you use the default or specify a low number for the monitoring server's well-known port.

### Tips

- Only one remote monitoring server in an LPAR can report to a given hub. If more than one remote monitoring server is configured in an LPAR, the hub to which each remote monitoring server reports must have a unique port number. Otherwise, connectivity problems might occur.
- If you change a port number for a hub monitoring server, you must reconfigure all the components that communicate with it. If you change a port number for a remote monitoring server, you must reconfigure all the agents that report to it. In a large enterprise, this can be a large task.

**Using the SKIP and COUNT options to control port number assignments:** For piped protocols (IP*, PIPE, but not IP*,UDP or SNA), you can use the SKIP and COUNT options to control the way port numbers are assigned to components. These options are specified on the KDE_TRANSPORT environment variable in the KppENV member of the &rthiev.&rte.RKANPARU library (where pp is the component code).

Example:

```
KDE_TRANSPORT="/\n    IP.PIPE PORT:1918 COUNT:1 SKIP:2\n    IP.UDP PORT:1918\n    SNA.PIPE PORT:135"
```

- The PORT option specifies the well-known port for the monitoring server.
• The COUNT:N option is the mechanism for reserving IP.PIPE ports for components that connect to the monitoring server. N is the number of IP.PIPE ports to reserve on the host system, in addition to the well-known port for the monitoring server.

For example, if the well-known port for the monitoring server is 1918, COUNT:3 starts the search at port 6014 (1918 + 1*4096). If the monitoring agent process cannot bind to port 6014, the algorithm tries port 10110 (1918 + 2*4096). If port 10110 is not available, the search goes to port 14206 (1918 + 3*4096). The monitoring agent is assigned to the first available port encountered in the search. The process fails to start if the search reaches the highest port number without a successful binding (port 14206 in this example).

Use the COUNT option to reserve ports for components that must be accessible from outside a firewall. Because these ports must be permitted at the firewall, the ports must be predictable.

• The SKIP:N option specifies the number of ports to skip when starting the search for an available port.

For example, if the well-known port for the monitoring server is 1918, SKIP:2 specifies to start the search at port 10110 (1918 + 2*4096), skipping ports 1918 and 6014 (1918 + 1*4096). The algorithm continues searching until it finds an available port.

Use the SKIP option for components that do not require access across a firewall.

**Using the POOL option to set a range of port numbers:** After the port allocation algorithm assigns a well-known port to each process, all subsequent ports allocated for connections between components are opaque ports; that is, any available port can be allocated for a connection. You can limit opaque port allocations to a specific range of ports by coding the POOL option with any protocol specified on the KDE_TRANSPORT environment variable.

The POOL option must specify a range of ports no smaller than 2 and no larger than 1024. POOL:1000-2023 is valid; POOL:1000-2024 is not. If more than 1024 ports are required in a pool for a specific protocol, you can code more than one POOL option, as in POOL:1000-2023 POOL:3000-4023.

**Network interfaces**

If your site runs more than one TCP/IP interface or network adapter on the same z/OS image, you can specify network interfaces to be used by monitoring servers and monitoring agents on a z/OS system. You specify the network interfaces in the IP communication protocol parameters for each component.

Before you begin configuring the monitoring server, decide whether you require a network interface list and, if so, which of the following values to specify for it:

• 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.
• A minus sign followed by a host name or IP address (-hostname) to use any interface except the one associated with hostname.

If you provide a value for the KDS_TEMS_TCP_KDEB_INTERFACELIST parameter in the PARMGEN configuration profile or for the Network interface list field in the Configuration Tool, the KDEB_INTERFACELIST environment variable is added to the KDSENV member of the &rhilev.&rte.RKANPARU library.
Important

- 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 z/OS image.
- In the default character set (locale 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.

See IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS for special considerations in specifying the KDEB_INTERFACELIST environment variable for a high-availability hub.

Disabling the HTTPS or HTTP server

If your environment uses an HTTP server but not an HTTPS server, or vice versa, you can prevent unnecessary error logging by specifying configuration parameters to disable the unused server. For a monitoring server, the HTTP or HTTPS server is used both for the Service Console and for the SOAP server. For a stand-alone monitoring agent, the HTTP or HTTPS server is used for the Service Console only.

To disable an HTTPS server, the HTTPS option in the KDE_TRANSPORT environment variable must be set to 0 (zero) in the Kpp ENV member of the &rhilev;&rte.RKANPARU library. To disable an HTTP server, the HTTP option in the KDE_TRANSPORT environment variable must be set to 0 (zero).

You can disable the HTTPS server by setting the value of the Kpp.X_KDE_TRANSPORT_HTTP_OPTIONS parameter to HTTPS:0 in the PARMGEN configuration profile, or use the Specify Nonstandard Parameters panel to add the HTTPS:0 option to the KDE_TRANSPORT environment variable. Example:

```
KDE_TRANSPORT______________________________
New Value: =HTTPS:0________________________________________
Old Value (if replacing): _______________________________________
Low-level data set qualifier: RKANPARU Member: KDSENV_
```

Similarly, you can disable the HTTP server by setting the value of the Kpp.X_KDE_TRANSPORT_HTTP_OPTIONS parameter to HTTP:0 in the PARMGEN configuration profile, or use the Specify Nonstandard Parameters panel to add the HTTP:0 option to the KDE_TRANSPORT environment variable.

Important: If you disable both the HTTP server and the HTTPS server in the KDSENV member (which contains environment variable definitions for the monitoring server), you disable the SOAP server. The SOAP server is required if you want to use the tacmd command-line utility to administer the self-describing agent feature.

For instructions on adding or changing parameter options so that they are not overridden when products are reinstalled or reconfigured, see "Adding, changing, or deleting a parameter in a runtime member" on page 98.
Decision 7: What types of runtime environments to set up

Your next planning task is to decide what types of runtime environments to set up for the components you plan to deploy on your z/OS images.

A runtime environment is a logical grouping of runtime libraries that are referenced by started tasks running on a z/OS image. When you configure monitoring servers and monitoring agents, you begin by defining a runtime environment of a certain type, which determines the number and types of runtime libraries required.

Table 5 summarizes the types of libraries created during installation and configuration of monitoring servers and monitoring agents on z/OS systems.

Table 5. Types of libraries

<table>
<thead>
<tr>
<th>Type of library</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime libraries</td>
<td>General term for libraries referenced by started task procedures.</td>
</tr>
<tr>
<td>Target libraries</td>
<td>SMP/E-maintained target libraries.</td>
</tr>
<tr>
<td>Abbreviated &amp;thilev.</td>
<td></td>
</tr>
<tr>
<td>Base libraries</td>
<td>Read-only runtime libraries that the configuration process does not alter and that are shareable between systems. These libraries physically exist in a full or base runtime environment, or as SMP/E target libraries (if a runtime environment shares with SMP/E). The base libraries can contain the actual data sets maintained by SMP/E, or a copy of them. Use a clone or copy of the SMP/E installation libraries for a production environment.</td>
</tr>
<tr>
<td>Abbreviated &amp;rhilev or &amp;rhilev.&amp;rte.</td>
<td></td>
</tr>
<tr>
<td>LPAR-specific libraries</td>
<td>Runtime libraries that are built during configuration to run on a specific logical partition (LPAR). These libraries contain the unique elements required for a particular LPAR and cannot be shared among z/OS images.</td>
</tr>
<tr>
<td>Abbreviated &amp;rhilev.&amp;rte.</td>
<td></td>
</tr>
</tbody>
</table>

Naming convention for runtime environment data sets

Each data set in a runtime library has a name composed of the following parts:

Table 6. Runtime environment data set naming convention

<table>
<thead>
<tr>
<th>Part of the name</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-level qualifier, VSAM or non-VSAM</td>
<td>26 bytes</td>
</tr>
<tr>
<td>Mid-level qualifier, which identifies the runtime environment</td>
<td>8 bytes</td>
</tr>
<tr>
<td>Low-level qualifier, which identifies the data set and matches the DDNAME</td>
<td>8 bytes</td>
</tr>
</tbody>
</table>

You set the high-level qualifiers for VSAM and non-VSAM data sets and the mid-level qualifier when you create a runtime environment. The low-level qualifier for each data set is provided by the configuration software and has the format
cKpfffffff

where
c is the data set class:

D for installation distribution libraries (DLIBs).
T for target libraries (TLIBs) that were installed by SMP/E.
R for runtime libraries.
pp is the 2-character code for the product or component.

ffff identifies the function of the data set; U as the final character identifies the data set as a user library. (Examples: CMD or CMDU for a commands data set, PAR or PARU for a parameters data set, and SAM or SAMU for a samples data set.)

For example, the data set IBM.RHILEV1.RTENAME1.RKANPARU has the high-level qualifier IBM.RHILEV1, the mid-level qualifier RTENAME1, and the low-level qualifier RKANPARU (which identifies the data set as a user library containing parameters for the component AN, the conglomerate component for the entire combination of products and components in the runtime environment).

**Naming convention for runtime environment jobs**

When you configure a runtime environment, you edit and submit several jobs. For a job that is submitted using the PARMGEN method, the naming convention is

KCIJcsss

where c = V if system variables are enabled and c = P if system variables are disabled and sss is the job name. For example, the KCIJVALO job allocates the runtime libraries with system variables, and the KCIJPALO job allocates the runtime libraries without system variables.

For job submitted by the Configuration Tool (ICAT) method, the member-naming convention for these jobs is

pp#tssss

where

- pp is the 2-character code for the product or component.
- t indicates the job type. The following are examples of some of the more common types:
  - 1 Build the runtime environment.
  - 2 Load the runtime environment.
  - 3 Create the runtime members.
  - 4 Register the monitoring agent with the monitoring server.
  - 5 Allocate additional runtime libraries job.
  - 6 Create special HFS and UNIX System Services related members.
  - 7 Rename the VTAM major node.
  - 8 Create a single VTAM major node.
  - D Delete the runtime environment.
  - L Process the LU6.2 logmode.
  - M Migrate or upgrade the runtime environment, a product, or a component.
  - P Create runtime members for the persistent data store.
  - Q Allocate and initialize the persistent data store files.
  - V Create the system variable parameter member.

- sssss is the unique JCL suffix identifying the jobs submitted for the runtime environment.

**Possible configurations using different types of runtime environments**

Table 7 on page 37 explains the types of runtime environments that you can create during product configuration.
<table>
<thead>
<tr>
<th>Type of runtime environment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full (self-contained) runtime environment</td>
<td>Runtime environment containing a full set of dedicated libraries, consisting of both LPAR-specific libraries and a copy of the SMP/E installation read-only base libraries eligible for sharing with other runtime environments. See <a href="#">Example 1. Full (self-contained) runtime environment</a> on page 38.</td>
</tr>
<tr>
<td>Base runtime environment</td>
<td>Runtime environment containing exclusively read-only, shareable base libraries, which are a subset of the libraries required for running TMS:Engine-based products. Therefore, a base runtime environment must be shared by another runtime environment. See <a href="#">Example 2. Base runtime environment</a> on page 39.</td>
</tr>
<tr>
<td>Sharing-with-full runtime environment</td>
<td>Runtime environment containing LPAR-specific libraries and referencing the base libraries configured in a full runtime environment. See <a href="#">Example 4. Sharing-with-full runtime environment</a> on page 42.</td>
</tr>
<tr>
<td>Sharing-with-SMP/E runtime environment</td>
<td>Runtime environment containing LPAR-specific libraries and referencing the libraries managed by SMP/E. See <a href="#">Example 5. Sharing-with-SMP/E runtime environment</a> on page 43.</td>
</tr>
</tbody>
</table>

The distinction among library types helps you to optimize your product environment. For example, by allocating common base libraries to a single runtime environment that can be shared by other runtime environments, you can substantially reduce the amount of disk space required and simplify the application of maintenance across z/OS images.

**Quick suggestions**

See the following suggested strategies for deciding the type of runtime environment to configure based on your requirements:

- If you plan to install monitoring agents on many z/OS images, you can get good results with a sharing-with-base or sharing-with-SMP/E type of runtime environment. See the following examples for considerations when using any type of sharing environment.
- If you want to test quickly, use a sharing-with-SMP/E type of runtime environment.
- If you want to test your configuration on an isolated test system, use a full, self-contained type of runtime environment.

The following five examples show different types of runtime environment configurations. The way you choose to set up your runtime environments depends on your site requirements and maintenance procedures.

**Tip**

The data set name (DSN) is composed of the high-level qualifier (&rhilev), followed by the mid-level qualifier (&rte), followed by the low-level qualifier. The field settings and library names shown are for illustrative purposes only.
Example 1. Full (self-contained) runtime environment

The full runtime environment contains all libraries required by a particular IBM product and is the easiest runtime environment to create. This type of runtime environment can be defined in any situation but is most suitable if at least one of the following statements is true:

- Your installation comprises only a single z/OS image.
- You want each z/OS image to be independent.
- You are creating a runtime environment for a specific combination of products that does not exist in any other runtime environment.

The following example represents a full runtime environment called RTE1 that is completely self-contained. All base libraries and LPAR-specific libraries are allocated in RTE1. The base libraries in a full runtime environment are a copy of the SMP/E installation libraries.

Name: RTE1
Type: Full
Hilev: PROD.CAN
Midlev: RTE1
Shares with: (none)

LPAR-specific library DD DSNAMES resolution:

```
//RKANPAR DD DISP=SHR,
//  DSN=PROD.CAN.RTE1.RKANPARU
//  DD DISP=SHR,
//  DSN=PROD.CAN.RTE1.RKANPAR
//RKANCMD DD DISP=SHR,
//  DSN=PROD.CAN.RTE1.RKANCMOU
//  DD DISP=SHR,
//  DSN=PROD.CAN.RTE1.RKANCMD
//STEPLIB DD DISP=SHR,
//  DSN=PROD.CAN.RTE1.RKANMODU
```

Base library DD DSNAMES resolution:

```
//RKANMOD DD DISP=SHR,
//  DSN=PROD.CAN.RTE1.RKANMOD
//RKANMODL DD DISP=SHR,
//  DSN=PROD.CAN.RTE1.RKANMODL
```

Figure 3 on page 39 illustrates a full runtime environment.
Figure 4 shows the way a full runtime environment can be expanded to more than one z/OS image. Each runtime environment is self-contained; the three runtime environments X, Y, and Z on systems A, B, and C do not share any libraries.

Example 2. Base runtime environment
The base runtime environment allocates shareable base libraries only. A base runtime environment must be used in conjunction with a sharing-with-base runtime environment (see "Example 3. Sharing-with-base runtime environment" on page 40) to provide the complete set of libraries required to run the installed
products. The sharing-with-base runtime environment must contain the same set or a subset of the products and components in the base runtime environment.

A base runtime environment is typically used when storage devices are shared or when product maintenance synchronization across systems is desired. Sharing base libraries avoids unnecessary duplication, saves disk space, and simplifies the application of maintenance to a common point.

The following example represents a base runtime environment called RTE2.

Name: RTE2
Type: Base
Hilev: COMMON.BASE
Midlev: RTE2 (Optional for a base runtime environment)
Shares with: (none)

LPAR-specific library DD DSNNAME resolution:
There are no LPAR-specific libraries in a base runtime environment.

Base library DD DSNNAME resolution:
//RKANMOD DD DISP=SHR,
// DSN=COMMON.BASE.RTE2.RKANMOD
//RKANMODL DD DISP=SHR,
// DSN=COMMON.BASE.RTE2.RKANMODL

Figure 5 illustrates a base runtime environment.

Note that while a mid-level qualifier is not required for a base runtime environment, it is considered a best practice to use one.

**Example 3. Sharing-with-base runtime environment**
The sharing-with-base configuration is a good choice for environments where storage devices are shared. Using the base runtime environment for common data sets, the sharing-with-base runtime environment contains only LPAR-specific libraries. The base runtime environment cannot contain the LPAR-specific libraries required to run the installed products. The sharing-with-base runtime environment must contain the same set or a subset of the products and components in the base runtime environment.
The configuration software resolves product configuration elements to point at the LPAR-specific libraries and the base runtime environment libraries as necessary.

The following example represents a sharing-with-base runtime environment called SHARRTE2, which obtains its base library information from the base runtime environment (RTE2).

**Name:** SHARRTE2  
**Type:** Sharing  
**Hilev:** E.F.G  
**Midlev:** SHARRTE2  
**Shares with:** Base RTE2

**LPAR-specific library DD DSNAME resolution:**

```plaintext
//RKANPAR DD DISP=SHR,  
// DSN=E.F.G.SHARRTE2.RKANPARU  
//  DD DISP=SHR,  
// DSN=E.F.G.SHARRTE2.RKANPAR  
//RKANCMD DD DISP=SHR,  
// DSN=E.F.G.SHARRTE2.RKANCMDU  
//  DD DISP=SHR,  
// DSN=E.F.G.SHARRTE2.RKANCMD  
//STEPLIB DD DISP=SHR,  
// DSN=E.F.G.SHARRTE2.RKANMODU
```

**Base library DD DSNAME resolution:**

```plaintext
//RKANMOD DD DISP=SHR,  
// DSN=COMMON.BASE.RTE2.RKANMOD  
//RKANMODL DD DISP=SHR,  
// DSN=COMMON.BASE.RTE2.RKANMODL
```

In typical product started tasks, the LPAR-specific libraries are concatenated ahead of the base libraries, as shown below:

```plaintext
//&PROCNAME PROC  
// SYS=SHARRTE2,  
// RHILEV='X.Y.Z',  
// BASEHILEV=COMMON.BASE.RTE2.R  
//STEPLIB DD DISP=SHR,  
// DSN=&RHILEV..&SYS..RKANMODU  
//  DD DISP=SHR,  
// DSN=&BASEHILEV.KANMODL  
//  DD DISP=SHR,  
// DSN=&BASEHILEV.KANMOD  
//RKANMODL DD DISP=SHR,  
// DSN=&RHILEV..&SYS..RKANMODU  
//  DD DISP=SHR,  
// DSN=&BASEHILEV.KANMODL  
//  DD DISP=SHR,  
// DSN=&BASEHILEV.KANMOD
```

*Figure 6 on page 42* illustrates a sharing-with-base runtime environment.
Example 4. Sharing-with-full runtime environment

The sharing-with-full runtime environment allocates LPAR-specific libraries only, and in this example, obtains its base library information from a full runtime environment. The sharing-with-base runtime environment must contain the same set or a subset of the products and components in the base runtime environment.

This configuration can also be used for environments where storage devices are shared, although the base/sharing pair is the preferred approach.

The following example represents a sharing-with-full runtime environment called SHARRTE1, which obtains its base library information from the full runtime environment (RTE1).

Name: SHARRTE1
Type: Sharing
Hilev: E.F.G
Midlev: SHARRTE1
Shares with: Full RTE1

LPAR-specific library DD DSNANE resolution:

//RKNPAR DD DISP=SHR,
// DSN=E.F.G.SHARRTE1.RKANPARU
// DD DISP=SHR,
// DSN=E.F.G.SHARRTE1.RKANPAR
//RKANCMD DD DISP=SHR,
// DSN=E.F.G.SHARRTE1.RKANCDU

Figure 6. Sharing-with-base runtime environment
Figure 7 illustrates a sharing-with-full runtime environment.

**Example 5. Sharing-with-SMP/E runtime environment**

The sharing-with-SMP/E runtime environment allocates LPAR-specific libraries only and obtains its base library information from target libraries managed by SMP/E.

Use the sharing-with-SMP/E configuration if at least one of the following statements is true:

- Space is limited on storage devices. This configuration method does not allocate base libraries in the runtime environment, thereby reducing storage requirements.
- You want to activate SMP/E-applied product maintenance immediately.

The following example represents a sharing-with-SMP/E runtime environment called SHARSMP, which obtains its base library information from SMP/E target libraries.

Name: SHARSMP  
Type: Sharing  
Hilev: E.F.G  
Midlev: SHARSMP  
Shares with: SMP/E target libraries using target Hilev (&thilev)  
INSTALL.SMPE
LPAR-specific library DD DSNAME resolution:

//RKANPAR DD DISP=SHR,
// DSN=E.F.G.SHARSMP.RKANPARU
// DD DISP=SHR,
// DSN=E.F.G.SHARSMP.RKANPAR
//RKANCMD DD DISP=SHR,
// DSN=E.F.G.SHARSMP.RKANCMDU
// DD DISP=SHR,
// DSN=E.F.G.SHARSMP.RKANCMD
//STEPLIB DD DISP=SHR,
// DSN=E.F.G.SHARSMP.RKANMODU
// DD DISP=SHR,
// DSN=E.F.G.SHARSMP.RKANMOD

Base library DD DSNAME resolution:

//RKANMOD DD DISP=SHR,
// DSN=INSTALL.SMPE.TKANMOD
//RKANMODL DD DISP=SHR,
// DSN=INSTALL.SMPE.TKANMODL

Figure 8 illustrates a sharing-with-SMP/E runtime environment.
Decision 8: What historical data to collect and how to manage it

Historical data collection is an optional feature that is enabled through the Tivoli Enterprise Portal. When you enable historical data collection, the monitoring agents are instructed to take data samples at a specified interval and store it. The collected data can be displayed in workspaces in the Tivoli Enterprise Portal, warehoused for in-depth analyses and long-term data reporting, and exported to third-party tools for reporting and analysis.

You configure and start historical data collection through the History Collection Configuration window of the Tivoli Enterprise Portal. In this window, you specify the attribute groups for which you want data to be collected, the interval for data collection, the location where you want the collected data to be stored (at the monitoring server or at the agent), and whether you want the data to be migrated to the Tivoli Data Warehouse.

On z/OS systems, short-term historical data is stored in data sets maintained by the persistent data store. When you configure monitoring servers and monitoring agents on a z/OS system, you allocate space for the persistent data store and specify the type of maintenance you want applied to the data sets.

Historical data collection consumes CPU and storage, so it is important to plan data collection carefully. Consider the following planning decisions before you begin to configure historical data collection and storage:

- “What data to collect and how often.”
- “Where to collect data” on page 46.
- “How much space to allocate” on page 46.
- “How to manage collected data” on page 47.

What data to collect and how often

The impact of historical data collection and warehousing depends on many factors, including collection interval, frequency of roll-off to the data warehouse, number and size of historical tables collected, amount of data, number of monitored resources, and system size. The IBM Tivoli Monitoring: Administrator’s Guide discusses the impact of historical data collection and warehousing on Tivoli Management Services components. The documentation for many OMEGAMON XE monitoring agents provides information about the space requirements and size of individual attribute tables, to help you estimate the impact of data collection.

Give careful consideration to what data you actually require. Historical data collection can be specified for individual monitoring servers, products, and attribute tables. Depending on your requirements, you can configure historical data collection for only a subset of attribute tables. Such a configuration can reduce storage and CPU consumption, particularly if you choose not to perform historical data collection for high-volume attribute tables or for attribute tables with many bytes per row (many attributes). Collect only the data that you plan to use in historical reports. Collect that data only as frequently as your enterprise requires.

The collection interval set in the History Collection Configuration window can be as short as a minute or as long as a day. The shorter the interval, the faster and larger the history files grow at the collection location. Short collection intervals also increase CPU consumption and network traffic. Do not set a one-minute collection interval unless your work requires it. If you require frequent collection of historical data, be sure to allocate extra space for the persistent data store. Insufficient allocation of space results in inability to view short-term data and can result in loss of historical data. Allocate enough space for 24 hours of short-term historical data at the location of the persistent data store.

Decisions about what data to collect always involve trade-offs between the usefulness of the data collected and the cost of collecting and managing the data.
**Where to collect data**

Historical data can be stored as short-term data either at the monitoring agent or at the monitoring server to which the agent reports. Where you decide to collect the data determines what steps you take to configure the persistent data store.

---

**Best practices**

Two OMEGAMON XE monitoring agents (OMEGAMON XE on z/OS and OMEGAMON XE for Storage on z/OS) must be configured within the address space of a monitoring server. Therefore, any historical data collection for these agents must also be stored at the monitoring server. For all other agents, whenever possible configure historical data to be collected and stored at the monitoring agent rather than at the monitoring server.

**Tip:** For best performance, avoid collecting historical data at the hub. See *IBM Tivoli Monitoring: Administrator's Guide* for details.

If an agent is configured to run in the monitoring server address space, like OMEGAMON XE on z/OS and OMEGAMON XE for Storage on z/OS, the persistent data store is configured in two locations:

- During configuration of the monitoring server, the generic persistent data store (RPDSGRP) is configured and the generic data sets (RGENHIS*) are allocated.
- During configuration of the OMEGAMON XE monitoring agent, product-specific historical data sets are allocated.

If a monitoring agent runs in its own address space (stand-alone), and you intend to collect historical data only at the location of the monitoring agent, you can configure the persistent data store during configuration of the agent only. However, to ensure that you have flexibility to collect data at the monitoring server at some later date, configure the persistent data store at the location of the monitoring server as well. If you have no monitoring server in the runtime environment, configuring the persistent data store for the monitoring agent allocates the generic data sets as well as the private data sets.

Data can be uploaded to the Tivoli Data Warehouse for long-term storage and reporting. Data warehousing is configured in the Historical Data Collection window of the Tivoli Enterprise Portal. See the *IBM Tivoli Monitoring: Installation and Setup Guide* for instructions on setting up the Tivoli Data Warehouse and the Warehouse Proxy and Summarization and Pruning agents. See the *IBM Tivoli Monitoring: Administrator's Guide* for instructions on configuring data warehousing.

**How much space to allocate**

The data written to the persistent data store is organized by tables (attribute groups), groups, and data sets. Each table is assigned to a group, which can have one or more tables and one or more data sets assigned to it. Normally, three data sets are assigned to each group.

When you configure the persistent data store, the persistent data store processing computes how much space is required for the group data store files and how much additional required space is required for overhead information. (Overhead information includes the product dictionary, table records, index records, and spare room for buffers that must be reserved for when the data set is full.) However, you might have to adjust the space allocated.

Ideally, you want to allocate enough space to hold 24 hours of data. Eventually, you can determine the correct amount of space by observing how often the maintenance procedures are running and adjusting space according. To help you make more specific calculations, the product documentation for the monitoring agents provides estimated space requirements or information about attribute tables. You might want to make your own calculations, based on site-specific factors: what types of monitoring agents are running, what resources are being monitored, how many resources, and so on.
You can also take a trial-and-error approach. To compute space requirements by trial and error, start historical collection for all resources you want to store in short-term history, and collect at least 25 hours of data. If you cannot draw a report in the Tivoli Enterprise Portal containing the last 24 hours of data for any attribute table, then you must allocate additional storage or data sets or both. You can override the persistent data store defaults to specify data store file placement and space allocation, and to set up maintenance for the data store.

When historical data is collected in product-specific data sets at the agent, allocate enough space so that you can reliably retrieve 24 hours of data for short-term history queries. Of the allocated data sets, one is always in use (Active), one is held aside empty to ensure immediate switchover when the in-use data set fills up, and the remainder are either Empty, Partially Full, or Full. With a default group count of 3 data sets, you have to size the total space so that one-third of the space is sufficient to hold 24 hours of data.

Increasing the group count (number of data sets used to provide the total number of cylinders of persistent data store space) is a way to reduce the total space required, by reducing the percentage of the total space required for the in-use and empty data sets. Increasing the group count to 6 or 8 dramatically reduces the total space required. (The maximum group count value is 36.) Another benefit of increasing the group count is that roll-off to the warehouse as well as short-term history queries perform better when the individual data sets are smaller. The overhead for more frequent data set switches caused by the increased group count (and presumably by executions of the maintenance procedures) is more than offset by the gains in efficiency.

When historical data is collected in generic history data sets at the monitoring server, the same considerations apply, but the data sets must be sized to contain the data from all of the agents that write to them.

**How to manage collected data**

If historical data collection is configured, provision must be made for handling the collected data. Without such provision, on distributed computers the history data files grow unchecked, using up valuable disk space. On z/OS systems, the data sets allocated to the persistent data store are emptied and overwritten.

On distributed systems, if warehousing of the data is configured, the files are automatically pruned after the data is inserted into the Tivoli Data Warehouse by the Warehouse Proxy agent. On z/OS systems, the persistent data store data sets are not pruned after data is inserted into the Tivoli Data Warehouse by the Warehouse Proxy agent. A record is written in the persistent data store to indicate where to begin exporting data to the Warehouse Proxy agent the next time, and the exported data is kept in the persistent data store until it is overwritten or deleted by other means.

Tivoli Management Services provides automatic maintenance for the data sets in the persistent data store. When a data set becomes full, the persistent data store selects an empty data set to make active. When the data set is active, the persistent data store checks to see whether any data sets are empty. If not, the persistent data store begins maintenance on the oldest data set. However, before it begins processing, it checks to see whether one or more of the following functions have been configured:

- **BACKUP** makes an exact copy of the data set being maintained.
- **EXPORT** writes the data to a flat file in an internal format that can be used by external programs to post-process the data. This function is also used for recovery when the persistent data store detects potential problems with the data.
- **EXTRACT** writes the data to a flat file in human-readable form suitable for loading into other database management systems (DBMS).

If no function has been specified, the data is deleted from the data set.

You specify which maintenance options are implemented when you configure the persistent data store. [Appendix C, "Maintaining the persistent data store," on page 185](#) provides more information about how the persistent data store operates and how maintenance is performed.
Decision 9: Which security options to enable

The security options you decide to employ can determine tasks you must complete in advance, such as arrangements you must make with security administrators or accounts you must set up. Your security decisions can also dictate certain choices during configuration, such as the selection of secure protocols when configuring communication between components.

Tivoli Management Services and the OMEGAMON XE monitoring agents offer several security options:

- Tasks to complete before enabling security
- IBM Tivoli OMEGAMON enhanced 3270 user interface security considerations
- "Secure communication between components" on page 49
- "Authorization and authentication of Tivoli Enterprise Portal users" on page 50
- "Authentication of OMEGAMON 3270 and OMEGAMON II (CUA) interface users" on page 50
- "Authentication of SOAP server users" on page 51
- "Authentication of IBM Tivoli Monitoring Service Console users" on page 51
- Appendix E, “Predefining and managing OMEGAMON started tasks,” on page 203

The information in the following sections is intended to help you decide what security you want to enforce. After you decide what security you want to enable, complete the tasks described in "Tasks to complete before enabling security".

Tasks to complete before enabling security

The first time you configure the hub monitoring server, do not enable security. Complete the following steps before you reconfigure the hub to enable security:

1. Configure all products and verify that they are operating correctly.
2. If you choose a third-party security package, verify that it is installed and configured correctly for your site.
3. Create user IDs in the Tivoli Enterprise Portal, and authorize the users to access resources.
4. Create the user IDs and passwords on the system hosting the hub monitoring server.

You do not have to define and authorize additional user IDs before you enable security, but you must define and authorize the sysadmin user ID.

As part of your preparation for deployment, determine which users require access to the Tivoli Enterprise Portal and which features, applications, and views the users must access.

Also determine which users you want to authorize to issue Take Action commands from the Tivoli Enterprise Portal. You can have the issuers of z/OS console commands authorized by Tivoli NetView for z/OS (see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS). Monitoring agents can have product-specific requirements for authentication. Consult the documentation for each monitoring agent for more information.

IBM Tivoli OMEGAMON enhanced 3270 user interface security considerations

The OMEGAMON enhanced 3270 user interface is a shared user interface providing data and function across multiple OMEGAMON XE monitoring agents. Authentication and authorization for users of the OMEGAMON enhanced 3270 user interface is provided by using the system authorization facility (SAF) interface. The existence of the SAF user ID and its validity are always checked. Three types of SAF authorization checks are also performed:

- Does the user have the authority to log on to this instance of the OMEGAMON enhanced 3270 user interface?
• Does the user have the authority to view the data from queries for specific types of data (attribute groups) on a specific managed system?
• Does the user have the authority to transmit a Take Action request to a specific managed system?

**Note:** The OMEGAMON enhanced 3270 user interface verifies only that the user has the authority to transmit Take Action commands to a specific managed system.

By default, when first installed, all users are allowed to log on to the OMEGAMON enhanced 3270 user interface and to view data from any managed system, but all Take Actions are denied. After security is configured, user permissions and the amount of security imposed are assigned by site administrators. All authentication or authorization failures are logged. All Take Action requests are logged.

Planning for security includes deciding the granularity you want to implement for protecting logon, queries and Take Action commands, as well as deciding which roles are granted access to the OMEGAMON enhanced 3270 user interface, and finally choosing or creating an SAF class that will contain the SAF resources. At a minimum, update the security settings to secure the Take Action function. Failure to correctly secure this powerful function of the OMEGAMON enhanced 3270 user interface might give all users full control to modify the managed system, including starting and stopping applications.

Determine who requires access to the OMEGAMON enhanced 3270 user interface, what information they may view and what Take Action commands they should have permission to invoke. Before security is configured in the environment, a security administrator must complete the following setup tasks:

1. Define an SAF general resource class
2. Define logon profiles to control access to the OMEGAMON enhanced 3270 user interface
3. Define Take Action profiles to control access to OMEGAMON enhanced 3270 user interface data actions
4. Define Query profiles to control access to OMEGAMON enhanced 3270 user interface data sources
5. Permit access to the profiles by appropriate personnel
6. Define a PASSTKT profile to allow the OMEGAMON enhanced 3270 user interface to propagate, identify, and perform authentication checking for query and action requests sent to the Tivoli Enterprise Monitoring Server

**Secure communication between components**

User IDs and passwords sent between Tivoli Management Services components are encrypted by default. To secure other communications, use SPIPE as the protocol when you configure communication between the Tivoli Enterprise Portal Server and the hub Tivoli Enterprise Monitoring Server, between hub and remote monitoring servers, and between agents and monitoring servers.

Two additional protocols are used for securing communications:

- HTTPS to retrieve files and Interoperable Object Reference (IOR). The integrated browser in the client provides HTTPS support on the client side; for the server, consider using a web server that supports HTTPS, such as the IBM HTTP Server. See *IBM Tivoli Monitoring: Installation and Setup Guide* for more information on using web servers.

  For information about disabling an HTTPS server, see "Disabling the HTTPS or HTTP server" on page 34.

- Internet Inter-ORB Protocol (IIOP) to secure the communications between the portal server and client. The IIOP uses Secure Socket Layer (SSL). This secure communication uses public key cryptography. SSL on z/OS systems requires the z/OS Communications Server. See *IBM Tivoli Monitoring: Installation and Setup Guide* for more information on using SSL for the distributed components.

IBM Tivoli Monitoring includes the Global Security Toolkit (GSKit) for SSL processing, which is also used in SPIPE and HTTPS. GSKit is installed by default with all distributed components, and its utilities are used to create and manage the encryption of data between components through the use of digital certificates.
On z/OS systems, GSKit is known as the Integrated Cryptographic Service Facility, or ICSF. If ICSF is not installed on the z/OS system, the monitoring server uses an alternative, less secure encryption scheme. Because both components must be using the same scheme, if the hub system does not use ICSF, you must configure the Tivoli Enterprise Portal to use the less secure scheme (EGG1) as well. For more information, see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.

A default certificate and key are provided with GSKit at installation. A stash file provides the database password for unattended operation. You can also use the key management facility (iKeyMan) in GSKit to generate your own certificates.

**Authorization and authentication of Tivoli Enterprise Portal users**

Access to the Tivoli Enterprise Portal (authorization) is controlled by user accounts (IDs) defined to the portal server. In addition to defining the user IDs that are authorized to log on to the Tivoli Enterprise Portal, these accounts define the permissions that determine the Tivoli Enterprise Portal features a user is authorized to see and use, the monitored applications the user is authorized to see, and the Navigator views (and the highest level within a view) the user can access. An initial sysadmin user ID with full administrator authority is provided during installation so you can log in to the Tivoli Enterprise Portal and add more user accounts. No password is required to log on to the Tivoli Enterprise Portal, unless user authentication is enabled.

Authentication can be enabled through either the hub Tivoli Enterprise Monitoring Server or the Tivoli Enterprise Portal Server. The hub Tivoli Enterprise Monitoring Server can be configured to authenticate, or validate, user IDs using either the local system registry or an external LDAP-enabled registry. The Tivoli Enterprise Portal Server can be configured to authenticate through an external LDAP registry.

Authentication by an external LDAP registry is not supported for a hub on z/OS systems. If the hub monitoring server is running on a z/OS system, you must configure RACF® or your system authorization facility (SAF) product to authenticate your Tivoli Enterprise Portal users. Alternatively, you can configure authentication through the Tivoli Enterprise Portal Server. If authentication is not enabled through either the monitoring server or the portal server, no authentication is performed and no password is required to log on to the Tivoli Enterprise Portal.

User IDs that have to make SOAP Server requests (including user IDs that issue CLI commands that invoke SOAP server methods) can be authenticated only through the hub monitoring server. User IDs that require the ability to share credentials with other web-enabled Tivoli applications (single sign-on capability, or SSO) must be authenticated through the portal server and mapped to unique user identifiers in an LDAP registry shared by all SSO-eligible Tivoli applications.

Do not enable user authentication before completing and testing at least a basic installation of Tivoli Management Services components and monitoring agents. For instructions on enabling authentication on a hub monitoring server on Windows, UNIX, and Linux operating systems, managing user accounts and permissions, or enabling the Tivoli Enterprise Portal Server for single sign-on, see the IBM Tivoli Monitoring: Administrator's Guide. For instructions on enabling authentication on a hub monitoring server on a z/OS system, see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.

**Authentication of OMEGAMON 3270 and OMEGAMON II (CUA) interface users**

You can provide security for the OMEGAMON 3270 and OMEGAMON II interfaces by using a combination of security types and implementations.

You must implement security at both the product level and the command level. Product-level security provides user ID and password validation to detect and prevent unauthorized access to the OMEGAMON...
II product, starting with the System Status panel. Command-level security prevents the unauthorized use of sensitive OMEGAMON for MVS commands from OMEGAMON II panels and by OMEGAMON users.

You can implement product-level and command-level security using either internal or external implementations, or a mixture. An internal implementation uses the security included with the product to control access. An external implementation uses another security package (RACF, CA-ACF2, or CA-TOP SECRET) to control access. To mix security choices at the product level and the command level, you can use, for example, RACF at the product level and internal security at the command level. If security is enabled on a z/OS hub monitoring server, you must use the same security implementation for the 3270 interfaces as is used for the hub.

See the configuration documentation for specific OMEGAMON XE monitoring agents for instructions on implementing security for 3270 interfaces.

**Authentication of SOAP server users**

User IDs that require access to the SOAP Server, including user IDs that issue commands that invoke SOAP methods, must be authenticated through the hub monitoring server. If user authentication is not enabled on the hub monitoring server, anyone can make requests to the SOAP Server. If user authentication is enabled on the hub, the SOAP Server honors requests only from user IDs and passwords authenticated by the local or external registry. If type of access is specified for specific users, requests from only those users for which access is specified are honored.

You can control access to the SOAP server in two ways:

- **You can control who is permitted to make requests by enabling user authentication on the hub monitoring server.**
  
  If the **Security: Validate User** option is not enabled, the SOAP server honors all requests regardless of the sender. If the **Security: Validate User** option on the hub monitoring server is enabled, the SOAP server honors requests only from users defined to the operating system or security authorization facility of the host of the monitoring server.

- **You can control what type of requests users are permitted to make by configuring the SOAP server.**

  **Important:** If you specify a specific type of access for any users, the SOAP server honors requests only from those users, regardless of whether **Security: Validate User** is enabled.

For information on configuring the security on the SOAP server, see *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS*.

**Authentication of IBM Tivoli Monitoring Service Console users**

The IBM Tivoli Monitoring Service Console enables you to read logs and turn on traces for remote product diagnostics and configuration. The Service Console performs user authentication using the native security facility of the host operating system. This means that if you use the Service Console on a z/OS system, your user ID and password are checked by the z/OS security facility (such as RACF/SAF). If you use the Service Console on Windows, the Windows workstation user ID and password are required for authentication. A password is always required to access the Service Console. Even if a user ID is allowed to log in to the operating system without a password, access to the Service Console is denied without a password. If necessary, create a password for the user ID that is being used to log in to the Service Console.

For more information about the Service Console, see *IBM Tivoli Monitoring: Troubleshooting Guide*.
What to do next

Note: If you are planning new started tasks and system automation, see the recommended naming conventions and critical planning information in Appendix E, “Predefining and managing OMEGAMON started tasks,” on page 203.

After you finish planning your deployment and installing components where desired, you can continue to the configuration tasks.

• For the PARMGEN configuration method, follow the instructions in Chapter 4, “Configuring products using the PARMGEN Workflow user interface,” on page 55.
• For the Configuration Tool (ICAT) method, follow the instructions in Chapter 5, “Configuring products with the Configuration Tool (ICAT),” on page 77.
Part 2. Configuring components on z/OS

The chapters in this part of the Common Planning and Configuration Guide address the tasks that you perform in configuring your products and components on z/OS systems:

- For the PARMGEN configuration method, follow the instructions in Chapter 4, “Configuring products using the PARMGEN Workflow user interface,” on page 55.
- For the Configuration Tool (ICAT) method, follow the instructions in Chapter 5, “Configuring products with the Configuration Tool (ICAT),” on page 77.

**Tips**

- Both the PARMGEN Workflow user interface and the Configuration Tool (ICAT) method are currently supported. For a first-time configuration, use the PARMGEN Workflow user interface for configuration of your products and components.
- If you are upgrading your products from a previous version, follow the instructions in IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Upgrade Guide, and refer to this part of the Common Planning and Configuration Guide for supporting information.

The PARMGEN method can be used for upgrading existing runtime environments that were created from product versions for which the PARMGEN configuration method was enabled. To upgrade from a product version lower than the first PARMGEN-enabled version, you must use the Configuration Tool. To determine the lowest PARMGEN-enabled version of each of your products, see the PARMGEN Technote at [http://www.ibm.com/support/docview.wss?uid=swg21417935](http://www.ibm.com/support/docview.wss?uid=swg21417935).
Chapter 4. Configuring products using the PARMGEN Workflow user interface

With the PARMGEN configuration method, you edit a comprehensive list of parameters for configuring all installed products and components. You then submit a series of jobs to create a complete runtime environment with the parameter values you specified.

The PARMGEN method can be used for creating new runtime environments and for upgrading existing environments that were created from product versions for which the PARMGEN configuration method was enabled. To upgrade from a product version lower than the first PARMGEN-enabled version, you must use the Configuration Tool (ICAT). To determine the lowest PARMGEN-enabled version of each of your products, see the PARMGEN Technote at http://www.ibm.com/support/docview.wss?uid=swg21417935.

If you have runtime environments that are already configured by the Configuration Tool (ICAT) method, a conversion tool is provided for using the existing parameter values in those runtime environments to set up initial values for new runtime environments to be configured by the PARMGEN method. After you use the PARMGEN method to configure new runtime environments, you cannot use the Configuration Tool (ICAT) to edit or maintain them.

The rest of this chapter gives basic instructions for using the PARMGEN method of configuration. A sample implementation, RTE=PLB1SP22, is presented in which the following parameters RTE_HILEV, RTE_VSAM_HILEV, and RTE_LIB_HILEV are set to this value: TSTEST.&userid. Upon completion of this chapter, you should be able to create a new runtime environment containing a monitoring server, monitoring agents, and other components such as the IBM Tivoli OMEGAMON enhanced 3270 user interface.

Review this chapter in conjunction with the product-specific Planning and Configuration guides to ensure that you understand and modify all required parameters for the environment that you want to create. The IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: PARMGEN Reference provides a set of typical deployment scenarios to help you create a runtime environment.

Prerequisite steps

Complete the following steps before you begin the configuration of your runtime environments using the PARMGEN method:

1. Ensure that you have completed SMP/E installation of the required levels of IBM Tivoli Management Services on z/OS components and OMEGAMON XE products for your monitoring environment. Check that all components that you want to configure are of the minimum supported level to support the PARMGEN method. For detailed information about supported product versions, see the PARMGEN Technote at http://www.ibm.com/support/docview.wss?uid=swg21417935.

2. Back up the &gbl_target_hilev.TKAN* target libraries that were installed by SMP/E. As a minimum, back up the &gbl_target_hilev.TKANMOD and &gbl_target_hilev.TKANCUS libraries.

3. Apply the latest HKCI310 PTF to ensure that you have the latest version of PARMGEN support. For the latest PTF number, see the Enablement Support section of the PARMGEN Technote at http://www.ibm.com/support/docview.wss?uid=swg21417935.

The HKCI310 PTF adds PARMGEN elements to the following target libraries:

- &gbl_target_hilev.TKANCMD
- &gbl_target_hilev.TKANCUS
- &gbl_target_hilev.TKANMOD
- &gbl_target_hilev.TKANPAR
- &gbl_target_hilev.TKANSAM
where &gbl_target_hilev is the high-level qualifier of the target libraries that were installed by SMP/E.

4. You must satisfy the following requirements to use the PARMGEN Workflow user interface:
   - A minimum region size of 6000K for the TSO user session
   - The PARMGEN Workflow user interface uses several standard TSO/E commands, and REXX and ISPF services. If you have modified or restricted access to any of these commands or services, the PARMGEN tool may not function correctly.

5. If you are planning to configure a monitoring server on a z/OS system and you enable the self-describing agent feature, ensure that you have created the appropriate folder structure on z/OS UNIX System Services (USS) under the Hierarchical File System (HFS) or zSeries File System (zFS).

   This step is required to store and process the provided self-describing files. The monitoring server also requires a Java runtime environment running under the IBM 31-bit or 64-bit JAVA SDK Version 5 (or higher), which is to be installed within the USS file system. A sample job to create a zFS file system is provided in the &gbl_target_hilev.TKANSAM(KCIUSSJB) member.

### Starting the PARMGEN Workflow user interface

The PARMGEN Workflow user interface assists you in setting up the PARMGEN work libraries for a runtime environment. Start the interface by executing the KCIR@PG1 member in the &gbl_target_hilev.TKANCUS library. The Welcome panel (KCIP@PG0) is displayed as shown in Figure 9.

---

**Figure 9. Welcome panel**

The Parameter Generator (PARMGEN) Workflow panel provides a convenient way for you to navigate through the PARMGEN steps to create a new runtime environment, clone an existing runtime environment, and maintain or upgrade a runtime environment. In the top part of the panel, you specify the parameters specific to the configuration and runtime environment that you are creating or modifying. The bottom part of the panel defines the tasks that must be performed in sequence to create a functioning runtime environment. It informs you about the status of each task and when the status was last changed.

If this configuration is new or you are returning to an existing configuration, you must specify all of the following parameters to identify the runtime environment that you are configuring:

**GBL_USER_JCL**

Specify the data set name of the PARMGEN global user JCL library. For planning purposes, this
data set is concatenated in the CONFIG DDNAME of product started tasks if the runtime environment is enabled for system variables. If the data set does not exist, you are prompted to correct the name or allocate the data set. The newly installed KCIJPCFG file is copied into this data set from the SMP/E &gbl_target_hilev.TKANSAM (target) library.

The PARMGEN global user JCL library is also where the PARMGEN process creates the system variables CONFIG profile. This is a required member for customization if you are using user-defined symbols or overriding system symbols resolved values when a runtime environment is enabled for system variables.

**RTE_PLIB_HILEV**

Specify the non-VSAM high-level qualifier that you want to use for the following PARMGEN work libraries:

- PARMGEN WCONFIG control library where CONFIG profiles and other members are stored by the KCIJPUP1 job.
- PARMGEN interim staging (IK*) libraries where PARMGEN product runtime members (template versions) are stored by the KCIJPUP1 job.
- PARMGEN work output (WK*) libraries where PARMGEN file-tailored runtime members and jobs are created by the $PARSE or $PARSES Eag.

**RTE_NAME**

Specify a unique name for the runtime environment that you are configuring. The name that you specify is appended to the RTE_PLIB_HILEV values to make each set of PARMGEN and runtime libraries unique.

---

**Step 1. Set up the PARMGEN work environment for a runtime environment**

The first step is to create the PARMGEN work environment. This step defines the libraries used throughout the runtime environment creation process, including the output libraries, and other parameters used in the generation of jobs that contribute to the runtime environment creation. It is also the process that you use to convert an existing Configuration Tool (ICAT)-defined runtime environment into a PARMGEN-defined runtime environment.

From the KCIP@PG0 main panel, select option 1 and press Enter. The Set Up PARMGEN Work Environment For An RTE (KCIP@PG1) panel is displayed as shown in Figure 10 on page 58.
Specify values for the required parameters as follows:

**RTE profile library and RTE member name**
Specify the runtime environment profile library and runtime environment member name, if applicable. The following options are available:
- If creating a brand new RTE, leave this field blank; or
- If creating another new RTE and you want to clone a PARMGEN-created RTE’s configured product set, specify the WCONFIG profile library and RTE member name to clone (for example: &hlq.&rte.WCONFIG(&clone_from)); or
- If upgrading an existing PARMGEN-maintained RTE, specify the WCONFIG profile library and RTE member name to upgrade; or
- If converting a Configuration Tool (ICAT)-created RTE to PARMGEN mode, specify the ICAT RTE Batch member location and RTE member (for example: &hlq.INSTJOBS(PLB1SP22))

**Install Job Generator (JOBGEN) output library**
Specify the Install Job Generator (JOBGEN) output library if you want PARMGEN to reuse CSI parameters from the JOBGEN repository:

**Jobcard data**
If a customized jobcard is already available, it is retrieved from the ISPF user profile pool. If it is not available and if you did not specify a JOBGEN output library, the jobcard is harvested from &gbl_target_hilev.TKANSAM SMP/E target library where the initial PARMGEN sample jobcard default is supplied. If you specified a JOBGEN output library, the jobcard information is harvested from that location. You can modify the retrieved data as needed. The customized jobcard is saved in the ISPF user profile pool and persists across ISPF sessions. When all required parameters are
entered, press Enter to move to step 2 as shown in Figure 11.

On this panel, specify the parameter values appropriate for your environment. In most cases, the PARMGEN configuration parameters on this panel are required. If you have specified the JOBGEN repository from the previous panel, the GBL_TARGET_HILEV, GBL_SYSDA, CSI_DSN and TARGET_ZONE are already specified for you.

GBL_INST_HILEV
Specify the high-level qualifier of the INSTJOBS ICAT installation jobs library if you have already created runtime environments with the Configuration Tool (ICAT) and you plan to convert those runtime environments into PARMGEN mode. The INSTJOBS library is where the ICAT Batch runtime environment member is stored. This member is used for the ICAT-to-PARMGEN conversion (via the KCIJPCNV conversion job). If you have not created runtime environments with ICAT, you are not required to provide a value for this parameter. By default, this field is prepopulated with the runtime environment profile library field value on the previous panel, if you have customized that field.

GBL_TARGET_HILEV
Specify the SMP/E high-level qualifier of the SMP/E target (TK*) libraries.

GBL_SYSDA_UNIT
Specify the non-VSAM disk UNIT for global work data sets.

CSI_DSN
Specify the data set name of the SMP/E global CSI where the product to be configured are installed.

TARGET_ZONE
Specify the name of the SMP/E CSI target zone where the product to be configured are installed.

When all required parameters have been entered, press Enter to move to step 3 as shown in Figure 12 on page 60.
On this panel, enter parameter values appropriate for your environment. Note that if you are using NONSMS-managed RTE_HILEV and RTE_VSAM_HILEV HLQs, the RTE_SMS_VOLUME, RTE_SMS_VSAM_VOLUME and RTE_SMS_UNIT values are required.

**RTE_SMS_PDSE_FLAG**
Specify whether or not (Y or N) partitioned data sets are to be allocated as PDSE. The default is Y.

**RTE_SMS_UNIT**
Specify the Non-VSAM disk UNIT type to be used for data set allocation.

**RTE_SMS_VOLUME**
Specify the Non-VSAM disk VOLSER to be used for data set allocation.

**RTE_SMS_MGMTCLAS**
Specify the Non-VSAM disk MGMTCLAS to be used for data set allocation.

**RTE_SMS_STORCLAS**
Specify the Non-VSAM disk STORCLAS to be used for data set allocation.

**RTE_SMS_VSAM_VOLUME**
Specify the VSAM disk VOLSER to be used for data set allocation.

**RTE_SMS_VSAM_MGMTCLAS**
Specify the VSAM disk MGMTCLAS to be used for data set allocation.

**RTE_SMS_VSAM_STORCLAS**
Specify the VSAM disk STORCLAS to be used for data set allocation.

**RTE_HILEV**
Specify the non-VSAM high-level qualifier that you want to use for the allocation of the non-VSAM production runtime (RK*) libraries. By default, the initial value for a brand new runtime environment to be created by using the PARMGEN process is the same value as the RTE_PLIB_HILEV value that you specified on the PARAMETER GENERATOR (PARMGEN) WORKFLOW - WELCOME panel.

**RTE_VSAM_HILEV**
Specify the VSAM high-level qualifier that you want to use for the allocation of the VSAM production runtime (RK*) libraries. By default, the initial value for a brand new runtime environment...
to be created by using the PARMGEN process is the same value as the RTE_PLIB_HILEV value you specified on the Parameter Generator (PARMGEN) Workflow - Welcome panel. When all required parameters have been entered, press Enter to move to step 4 as shown in Figure 13.

This panel shown in Figure 13 displays information about the SMP/E environment and the products available for configuration. Review the SMP/E message traffic before proceeding. These products are installed in the SMP/E CSI environment provided, and are available for configuration and upgrade into the runtime environment. The first group of messages are produced by a routine that extracts information from the CSI target zone. The CSI dsname and target zone name are listed.

If you specified an RTE CONFIG profile on the Set Up PARMGEN Work Environment For An RTE (1 OF 3) panel either as a Configuration Tool (ICAT) Batch parameter member to convert from &shilev:INSTJOBS or as an existing PARMGEN WCONFIG profile, the following messages may display: Active, installed components configured in the RTE profile: nn. This group of messages is a list and count of active (not superseded) FMIDs installed in the specified target zone. Press Enter to move to step 5 as shown in Figure 14 on page 62.
This panel lists the products installed in the CSI target zone and are therefore available for configuration. The products are listed by product code (Kpp) and product name/version. See Appendix D, “Product codes,” on page 201 for the complete list of product codes. If the list exceeds screen size, you can scroll down.

- If you want to exclude any of the products from the runtime environment that you are configuring, place an X before the product codes. In the example shown in Figure 14, all the products have been excluded except for KCS (OMEGAMON XE for CICS on z/OS), KDO (Tivoli Decision Support for z/OS), KDS (Tivoli Enterprise Monitoring Server), KGW (OMEGAMON XE for CICS TG on z/OS V510), KHI (OMEGAMON z/OS Management Console V410), KI5 (Tivoli OMEGAMON XE for IMS on z/OS V420), and KM5 (OMEGAMON XE on z/OS V510).

- If you specified an RTE CONFIG profile on the Set up PARMGEN Work Environment For An RTE (1 of 3) panel either as a Configuration Tool (ICAT) Batch parameter member to convert from &shilev.INSTJOBS or as an existing PARMGEN WCONFIG profile, any products already configured within that member are flagged with an asterisk following the product code. In the example shown in Figure 14, KD5* indicates that the OMEGAMON XE for DB2 Performance Expert or the OMEGAMON XE for DB2 Performance Monitor is currently configured within this runtime environment.

- Additionally, if an existing RTE CONFIG profile has been specified, the first line in the list of products specifies ALL as the product code. Press Enter to generate the GBL_USER_JCL(KCIJPCFG) file-tailored job. Review this job and submit it.

### Step 2. Review the PARMGEN job index

On the KCIP@PG0 main panel, note the job step indicator on 1> to denote that this is the last step you performed. Also, note the updated Status / Date on the KCIJPCFG option 1. Next, select option 2. $JOBINDEX to review the job index for planning purposes.

When you have completed the review, press F3 to complete this step.
Step 3. Update the interim libraries and create profiles

The job generated and submitted in this step provides two purposes. First, it populates (or updates if this is an existing runtime environment) the IK* staging libraries with the product-specific PARMGEN elements from the SMP/E target libraries. Second, it prepares the various PARMGEN elements based on the user-defined customizations made in step 1 (the KCIJPCFG job).

From the KCIP@PG0 main panel, select option 3 and submit the WCONFIG(KCIJPUP1) job. On completion, you should receive a good return code.

Step 4. Convert a Configuration Tool (ICAT) runtime environment

Batch member (Optional)

Note: This step is applicable only if you are converting an existing runtime environment created using the Configuration Tool (ICAT).

In this step, you take the properties and parameters from an existing runtime environment created using the Configuration Tool (ICAT) and run a job to convert them so that the settings are applied and serve as input to the

- $PARSE (if the runtime environment is not enabled for system variables); or
- $PARSESV (if the runtime environment is enabled for system variables) file-tailoring job within the new PARMGEN-configured runtime environment

Note that once you have converted the runtime environment to PARMGEN, you cannot use the Configuration Tool (ICAT) to manage the runtime environment any more. Prior to running this conversion job, you must create a batch mode parameter member with the existing Configuration Tool (ICAT)-configured runtime environment. See Chapter 7, “Using the Configuration Tool (ICAT) batch mode to replicate a configured environment,” on page 121 for details on creating a batch mode parameter member.

From the KCIP@PG0 main panel, select option 3 and then submit the WCONFIG(KCIJPUP1) job. On completion, you should receive a good return code.

Specify the runtime environments to convert

The PLBIN DD card specifies the RTE() parameter, which contains the list of runtime environment members that you want to convert from Configuration Tool (ICAT) to PARMGEN. By default, the KCIJPCFG job is listed, that is, the name of the runtime environment that you specified in “Step 1. Set up the PARMGEN work environment for a runtime environment” on page 57 If you must add additional runtime environments to convert, include them in the parameter as follows:

```
//PLBIN DD *
RTE(PLB1SP22 &rte_name2 &rte_name3 &rte_name4 &rte_name5) +
RTE(&rte_name6 &rte_name7 &rte_name8 &rte_name9 &rte_name10)
```

where &rte_name= the names of additional runtime environments to convert.

All batch mode parameter members to be converted should be located in the data set specified in the INDSN parameter under the SYSTSIN DD card.

Customizing the parameters

In addition to the data sets used for the input and output of this conversion process, the SYSTSIN DD card has the following customizable parameters:

- **REPLACE**
  Specifies if you want to replace the member in OUTDSN. The default is REPLACE.
OPT(xxx)

Specifies different options to the KCIRPLBC program supporting the following keywords in place of xxx:

- STRIP: if specified, any comment record or variable with no value will be stripped from the converted member.
- EXTRA: if specified, LOOP table data is written to SYSTSPRT.

This information identifies the tables in the converted member.

**Step 5. Customize PARMGEN configuration profiles**

In this step, you customize the parameters and settings for each of the components that you are deploying in this runtime environment, which may include a Tivoli Enterprise Monitoring Server, OMEGAMON enhanced 3270 user interface, and any OMEGAMON monitoring agents. Complete this step in conjunction with the product-specific Planning and Configuration guides and PARMGEN Reference guide, if available, to help you configure your components correctly.

Default values are provided for all required parameters and some optional ones. If you do not want to customize these parameters, and you do not want to enable optional features, you can complete the configuration by accepting these defaults. Alternatively, you can specify custom values. You can also specify custom values for optional parameters that have no defaults.

From the KCIP@PG0 main panel, select option 5 and press Enter. The Customize PARMGEN Configuration Profile Members (KCIP@PG6) panel is displayed as shown in **Figure 15**.

```
KCIP@PG6 ------ CUSTOMIZE PARMGEN CONFIGURATION PROFILE MEMBERS ----------
OPTION ===>
(Required)* Select option 1 to customize the PLBISP22 RTE LPAR profile
  1. PLBISP22 RTE LPAR CONFIG profile in WCONFIG
(Conditional)* Select option 2 and/or 3 if applicable to this RTE:
  2. $GBL$USR Global parameters CONFIG profile in WCONFIG
     (Required if this is not an ISAT-to-PARMGEN conversion)
  3. PLBISP22 System Variables CONFIG profile in GBL_USER_JCL
     (TSTEST.&userid.PARMGEN.JCL)
     (Required if using user-defined symbols or overriding system symbols' resolved values - see F1=Help)
*Note: The PARMGEN configuration profiles above are preserved (initially created by KCIJPUP1 job).

(Reference) IBM-supplied default profiles (refreshed by KCIJPUP1 job):
  4. $CFG$IBM IBM default RTE LPAR CONFIG profile in WCONFIG
  5. $GBL$IBM IBM default Global parameters CONFIG profile in WCONFIG
  6. $SYSIN SPARSE/SPARSESYS SYSIN controls for processing which:
     - CONFIG profiles (CONFIG MEMBER=&config_profile)
     - runtime members (SELECT MEMBER=(*,&mbr1,&mbr2??))
     to (re)create from PARMGEN IK*-to-WK* output libraries.
Enter=Next F1=Help F3=End/Cancel
```

**Figure 15. Customize PARMGEN configuration profile members**

Part of the configuration includes several substeps. The substeps that you must follow are dependent on the runtime environment that you are creating and the amount of customization you want to make to default parameters. Sub step 1 is a required step for all configurations. The remaining description of this section describes a basic set of customizations users may typically want to make to their runtime environment.
Tips

When editing the PARMGEN configuration profiles, you may have a large number of parameters to review. It may be useful, therefore, to EXCLUDE certain PARMGEN parameters while you focus on a particular configuration task. For example, EXCLUDE all PARMGEN parameters that are for started tasks. Then FIND all occurrences of these parameters so you can quickly do a CHANGE ALL command. Create an EXCLUDE-FIND (XF) macro for this task. See the PARMGEN help panels for more information.

Update the USER PROLOG (Optional)

The profile provides a section to log the changes you make within this member for future reference. Perform a FIND on USER PROLOG to locate this section and update it accordingly. Figure 16 shows an example of this section.

Verify the list of products to be configured within this runtime environment

Review the list of products that will be configured as part of this runtime environment. This list should match with the list of products that you defined earlier in the process. If any products are added or removed from this runtime environment after initial configuration, update this section accordingly. Figure 17 on page 66 shows an example of this section.
Update names of started tasks for all configured products

Perform an EXCLUDE FIND to list all the started task names for the products to be configured within this runtime environment. Figure 18 shows an example of this section.

Figure 17. Verify the list of products in the runtime environment

Figure 18. Update names of started tasks
In the example shown in Figure 18 on page 66, the names for all STCs are defined by using the default prefix (CANS*). Update these values to the naming convention appropriated for your environment.

**Update the TCP/IP port values used across the runtime environment**

If you are configuring the runtime environment with the default protocols of IP and IP.PIPE and plan to use a different port number from the default value (1918), you must update all occurrences within the profile. Perform an EXCLUDE FIND on _PORT to list all the TCP/IP port values to be used by the products within this runtime environment to communicate with each other. Update all port numbers to the value that you want to use. The ports for all monitoring agents and any monitoring servers must be the same for communication to be successful. Figure 19 shows an example of this section, where nnnnn is the port number that you choose to use.

![Update TCP/IP port values](image)

**Update the global VTAM major node**

Perform a FIND on RTE_VTAM_GBL_MAJOR_NODE to find the parameter for the global VTAM major node and update the value according to the conventions in your environment.

RTE_VTAM_GBL_MAJOR_NODE KCANDLE1

**Update VTAM node and application definitions**

If your environment will be configured to use the SNA protocol, perform an EXCLUDE FIND on _VTAM_APPL to find all the parameters associated for VTAM node and application definitions for all products within this runtime environment and update the values accordingly. Figure 20 on page 68 shows an example of this section.
Enable APF-authorization statements (Optional)

PARMGEN provides the ability to generate the required APF-authorization commands for all libraries concatenated within the STEPLIB and RKANMODL DD names for all generated STCs. To include these comments within the generated STCs, perform a FIND on RTE_X_STC_INAPF_INCLUDE_FLAG and update the value of the parameter to Y. If you do not enable this parameter, you must manually APF-authorize all libraries required as described in Chapter 6, "Completing the configuration," on page 103. Figure 21 on page 69 shows an example of this section.
Update the UNIX System Services (USS) directory

If any of the products that you are configuring require any UNIX System Services (USS) directories running on either the Hierarchical File System (HFS) or on the zSeries File System (zFS), or if you are planning to enable the self-describing agent feature, perform a FIND on the RTE_USS_RTE parameter and update the value to the directory to be used.

RTE_USS_RTEDIR "/rtehome"

Enable the self-describing agent feature at the Tivoli Enterprise Monitoring Server

If you are configuring a monitoring server as part of this runtime environment and are planning to enable the self-describing agent feature, check the value of the KDS_KMS_SDA parameter. By default, the self-describing agent feature is enabled within a remote monitoring server and any monitoring agent that provides self-describing support but the feature is disabled at the hub monitoring server. Update the value of KDS_KMS_SDA to Y to enable the self-describing agent feature at the hub monitoring server in this runtime environment. Figure 22 shows an example of this section.

Update any SMS-related values

If you must specify any SMS-related parameters, perform an EXCLUDE FIND on _STORCLAS or _MGMTCLAS to review the current settings and update the values accordingly. Figure 23 on page 70 shows an example of this section.
Configure a Tivoli Enterprise Monitoring Server

If you are configuring a Tivoli Enterprise Monitoring Server as part of this runtime environment, either as a hub or remote, the parameters that affect this component use the prefix KDS within the configuration profile. For more information about additional parameters to update, see the IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: PARMGEN Reference and the IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.

Configure the IBM Tivoli OMEGAMON enhanced 3270 user interface

If you are configuring an OMEGAMON XE monitoring agent that supports the OMEGAMON enhanced 3270 user interface as part of this runtime environment, you can choose to also create an instance of the OMEGAMON enhanced 3270 user interface. The parameters that affect this component use the prefix KOB within the configuration profile. Review these parameters and update the values accordingly. Figure 24 shows an example of this section.

Configure OMEGAMON monitoring agents and other components

Before completing all the updates to the PARMGEN profile, review and update, if needed, all remaining parameters specific to the various OMEGAMON monitoring agents and any other components that are
configured within this runtime environment. Component-specific parameters are prefixed with Kpp (where pp identifies the individual product). See Appendix D, “Product codes,” on page 201 for a full list of product codes. Review the product-specific Planning and Configuration guides and PARMGEN Reference guide, if available, to identify any further changes needed.

Update global parameters
After updating all settings within the WCONFIG(%RTE_NAME%) member, the next step is to review and update the global parameters that will be used by this runtime environment and any other runtime environment that you may configure. This step is required if you are not performing a Configuration Tool (ICAT)-to-PARMGEN conversion. If this situation applies, select option 2 from the CUSTOMIZE PARMGEN CONFIGURATION PROFILE MEMBERS panel. The WCONFIG($GBL$USR) member is displayed. This member contains all the parameters that define the global system libraries to be used by all components. Review and update, as appropriate, the parameter values contained in this member. All global parameters are prefixed GBL_.

The key parameters to check are the common system library values:

- GBL_DSN_SYS1_PARMLIB "SYS1.PARMLIB"
- GBL_DSN_SYS1_PROCLIB "SYS1.PROCLIB"
- GBL_DSN_SYS1_SAXREXEC "SYS1.SAXREXEC"
- GBL_DSN_SYS1_VTAMLIB "SYS1.VTAMLIB"
- GBL_DSN_SYS1_VTAMLST "SYS1.VTAMLST"

Health check configuration value: this library is used to set up the health check elements. By default, two health checks are provided (for STC and APF authorization):

- GBL_DSN_HZSPROC_LOADLIB "USER.LOADLIB"

If you are enabling the ITM Password Encryption (KAES256) across the enterprise, you must set the ICSF system library:

- GBL_DSN_CSF_SCSFMOD0 "CSF.SCSFMOD0"

If any of the components that you are configuring within this runtime environment require Java support or if you are planning to enable the self-describing agent feature, a valid path to a Java installation on HFS or zFS must be specified. Note that /bin is automatically appended:

- GBL_HFS_JAVA_DIR1 /usr/lpp/java/IBM/J6.0"

Also, if you are enabling the self-describing agent feature or any other function that uses z/OS UNIX System Services, check the value of the CLIST/EXEC library name. This EXEC library is used during the creation of USS directories running on either the Hierarchical File System (HFS) or on the zSeries File System (zFS):

- GBL_DSN_SYS1_SBPXEXEC "SYS1.SBPXEXEC"

If a monitoring server is using any of the IP.UDP-related or IP.PIPE-related communication protocols for connection, but the IP domain name resolution is not fully configured on the z/OS system, the SYSTCPD statement must be supported by the monitoring server and all monitoring agents that report to it. SYSTCPD explicitly identifies which data set to use to obtain the parameters defined by TCPIP.DATA when no GLOBALTCPIPDATA statement is configured. To support SYSTCPD, uncomment and set the following parameter:

- GBL_DSN_TCP_SYSTCPD_TCPIPDATA "TCPIP.SEZAINST"

The name of the SYSTCPD data set is installation-specific. Get the correct specification from your network administrator. Also ensure the following flags are enabled within the LPAR config profile:

- KAG_X_STC_SYSTCPD_INCLUDE_FLAG Y (for monitoring agent support)
- KDS_X_STC_SYSTCPD_INCLUDE_FLAG Y (for monitoring server support)
When all changes have been made, save the changes and return to the main PARMGEN Workflow user interface menu.
Step 6. Validate the PARMGEN profile parameter values

This step generates a job to validate all the parameters generated and edited in the previous step. From the KCIP@PG0 main panel, select option 6. KCIPVAL and press Enter. Submit the generated KCIPVAL job. If there are errors highlighted after this job finishes, return to the previous step and correct the parameter values.

Step 7. Create the runtime environment members and jobs

This step generates some jobs to parse the profile parameters detailed in previous steps. The jobs will then generate the runtime environment members and jobs required to complete the PARMGEN setup. From the KCIP@PG0 main panel, select option 7 and press Enter. The Submit $PARSE Batch Jobs To Complete PARMGEN Setup (KCIP@PRS) panel is displayed. [Figure 25] shows an example of this section.

```
KCIP@PRS ---- SUBMIT $PARSE BATCH JOBS TO COMPLETE PARMGEN SETUP ------------
OPTION ===> 1          SCROLL ===> CSR

Select option 1 to SUBMIT the full $PARSE job in WCONFIG.
Alternatively, select other options to SUBMIT $PARSE jobs individually.
Press F1=Help for additional considerations when selecting options 2-5.

Note: Enter ns (1s-5s) for detailed task status. Status Date

1. $PARSE Composite $PARSE job
2. $PARSECM IKANCMDU/WKANCMDU $PARSE job
3. $PARSEPR IKANPARU/WKANPARU $PARSE job
4. $PARSERM IKANSAMU/WKANSAMU $PARSE job
5. $PARSEDV Generate listing of symbolics

Enter=Next F1=Help F3=End/Cancel
```

Figure 25. Create the runtime environment members and jobs

You can either choose option 1 to submit the full set of parsing actions within a single job, or you can submit each action individually in turn by choosing options 2 through 5. You may want to follow the latter approach if you want to create (or recreate) only certain runtime members.

Step 8. Submit batch jobs to complete the PARMGEN setup

The final step within the PARMGEN Workflow user interface provides a series of batch jobs to submit that will create the runtime environment that you have defined in the previous steps. From the KCIP@PG0 main panel, select option 8 and press Enter. The Submit Batch Jobs To Complete PARMGEN Setup (KCIP@SUB) panel is displayed. [Figure 26 on page 74] shows an example of this section.
You can either choose option 1 to submit the full set of actions within a single job, or you can submit each action individually in turn by choosing options 2 through 12. Note that some options, such as USS system setup, may require specific user privileges to make changes to the file system. Because the user who has been working through PARMGEN may not have the correct authorization in place, specific jobs can be excluded and submitted by a user who does have the correct authorization.

If you decide to submit all the jobs automatically, submit the composite KCIJVSUB job (if system variables are enabled) or KCIJPSUB job (if system variables are disabled) in the WKANSAMU library. Otherwise, submit the jobs individually.

In the following list of batch job names, c = V if system variables are enabled and c = P if system variables are disabled in the WKANSAMU library. Otherwise, submit the jobs individually.

In the following list of batch job names, c = V if system variables are enabled and c = P if system variables are disabled. For example, the KCIJVALO job allocates the runtime libraries with system variables, and the KCIJPALO job allocates the runtime libraries without system variables.

**KCIJcALO**
This required job allocates the RK* runtime libraries for all the products and components in the runtime environment.

**KCIJcLOD**
This required job copies members of the target libraries that were installed by SMP/E to the read-only RK* libraries. It also deletes any runtime members from the different libraries based on the product versions' FMID requirements. For example, if the latest version of the product removes or renames any SMP/E elements.

Specifically, the BUILDEX/DELRUN step bypasses delete requests for elements that may already have been deleted from a previous run of the KCIJcLOD RTE load job. Additionally, if any of the libraries listed in the DELRUN DDNAME is part of the system link libraries, or the library may be in use, the KCIJcLOD RTE load job may not be able to delete these elements if the libraries are enqueued. If either of these situations are true, run the BUILDEX/DELRUN step when those libraries are available.

**KCIJcSEC**
This job is required if the product-specific IBM-supplied security exit or input must be customized. The job creates security-related members (load modules, encryption key, and other elements) based on the product security requirements.

---

**Figure 26. Submit batch jobs to complete PARMGEN setup**

You can either choose option 1 to submit the full set of actions within a single job, or you can submit each action individually in turn by choosing options 2 through 12. Note that some options, such as USS system setup, may require specific user privileges to make changes to the file system. Because the user who has been working through PARMGEN may not have the correct authorization in place, specific jobs can be excluded and submitted by a user who does have the correct authorization.

If you decide to submit all the jobs automatically, submit the composite KCIJVSUB job (if system variables are enabled) or KCIJPSUB job (if system variables are disabled) in the WKANSAMU library. Otherwise, submit the jobs individually.

In the following list of batch job names, c = V if system variables are enabled and c = P if system variables are disabled. For example, the KCIJVALO job allocates the runtime libraries with system variables, and the KCIJPALO job allocates the runtime libraries without system variables.

**KCIJcALO**
This required job allocates the RK* runtime libraries for all the products and components in the runtime environment.

**KCIJcLOD**
This required job copies members of the target libraries that were installed by SMP/E to the read-only RK* libraries. It also deletes any runtime members from the different libraries based on the product versions' FMID requirements. For example, if the latest version of the product removes or renames any SMP/E elements.

Specifically, the BUILDEX/DELRUN step bypasses delete requests for elements that may already have been deleted from a previous run of the KCIJcLOD RTE load job. Additionally, if any of the libraries listed in the DELRUN DDNAME is part of the system link libraries, or the library may be in use, the KCIJcLOD RTE load job may not be able to delete these elements if the libraries are enqueued. If either of these situations are true, run the BUILDEX/DELRUN step when those libraries are available.

**KCIJcSEC**
This job is required if the product-specific IBM-supplied security exit or input must be customized. The job creates security-related members (load modules, encryption key, and other elements) based on the product security requirements.
Review the KppSUPDI "Modify Classic command table" security steps. The KppJPSC0 input members to the composite KCIJPSEC security job point SYSIN DD to WKANSAMU(KppSUPDI) by default. If you must make further changes to the sample exit, modify the SYSIN DD accordingly. Also see the RTE_X_SECURITY_EXIT_LIB parameter in the PGN3MVSE CONFIG LPAR profile.

**KCIJcUSP**
This job is required if you are configuring at least one product that requires UNIX System Services (USS). The job creates the USS-related members in the RKANDATV runtime library for use in the composite KCIJcUSS job.

**Tip:** This job is required if you enabled the self-describing agent feature.

**KCIJcUSS**
This job is required if you are configuring at least one product that requires USS. The job creates the Hierarchical File System (HFS) or the zSeries File System (zFS) directories and subdirectories, and copies files to HFS or zFS.

**Tip:** This job is required if you enabled the self-describing agent feature.

**KCIJcSYS**
This job copies the product started tasks, VTAM major node members, and health check elements for the products and components into system libraries, and assembles and links product modules into system libraries. The job requires write access to system libraries.

**KCIJcLNK**
The job assembles and links elements into the SYSLMOD RKANMOD* load library.

**KCIJcUPV**
If system variables are enabled (RTE_SYSV_SYSVAR_FLAG = Y), this job must be submitted in the target LPAR where the symbolics are resolved. This job populates variable-named members contained in the application-specific KppJPUPB composite IEBUPDTE members in the WK* work output libraries. See the KCIP@SUB help panels for more information.

**KCIJcCPY**
This job backs up the RK* runtime libraries and the WK* work output libraries. If you run this job individually (rather than running the composite job to submit all the batch jobs), run the job three times:

1. Clone the PARMLIB interim staging libraries (IK*).
2. Clone the PARMLIB work output libraries (WK*).
3. Clone the existing production runtime user libraries (RK*).

**KCIJcW2R**
This job copies the WK* work output libraries to the respective production RK* runtime libraries. If you do not run this job, you must copy the work output libraries to the runtime libraries by some other method, following your normal change control process.

**KCIJcIVP**
This required job verifies that all the required runtime data sets, members, and configuration jobs for the runtime environment were created, and that the jobs were executed successfully.

On completion of these jobs, review the output of the KCIJcIVP configuration verification job:

1. Review the $IVPRPT report, which is stored in the WCONFIG library. This report shows the results of each KCIJP* job that was run to configure the runtime environment. In the following example, the report shows that the KCIJPLOD job had an ABEND E37 out-of-space condition in one or more of the libraries loaded by the job:

<table>
<thead>
<tr>
<th>JOB</th>
<th>STATUS</th>
<th>JOBNAME</th>
<th>JOB#</th>
<th>DATE</th>
<th>TIME</th>
<th>HI-CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCIJPCFG OK</td>
<td>CCAP1$JP</td>
<td>J29242</td>
<td>10.056</td>
<td>22:46:42</td>
<td>00000</td>
<td></td>
</tr>
</tbody>
</table>
2. Review the WSUPERC SYSTSPRT report, which is stored in the WSUPERC sequential library. This report compares the PARMGEN WK* work output libraries with the production RK* runtime libraries. The PARMLIB WK* work output libraries are the data sets created and populated by the $PARSE job. In the following example, two files in the new WKANCMDU data set do not have equivalents in the existing RKANCMDU data set:

<table>
<thead>
<tr>
<th>COUNTS</th>
<th>WKANCMDU</th>
<th>RKANCMDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL MEMBER(S) PROCESSED AS A PDS</td>
<td>48</td>
<td>47</td>
</tr>
<tr>
<td>TOTAL MEMBER(S) PROCESSED HAD CHANGES</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>TOTAL MEMBER(S) PROCESSED HAD NO CHANGES</td>
<td>47</td>
<td>46</td>
</tr>
<tr>
<td>TOTAL NEW FILE MEMBER(S) NOT PAIRED</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL OLD FILE MEMBER(S) NOT PAIRED</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ISRSUPC - MVS/PDF FILE/LINE/WORD/BYTE/SFOR COMPARE UTILITY- ISPF</td>
<td>NEW: &amp;rhilev.&amp;rte.WKANCMDU OLD: &amp;rhilev.&amp;rte.RKANCMDU</td>
<td>MEMBER SUMMARY LISTING (LINE COMPARE)</td>
</tr>
<tr>
<td>NON-PAIRED NEW FILE MEMBERS</td>
<td>KC2OPS01</td>
<td></td>
</tr>
<tr>
<td>NON-PAIRED OLD FILE MEMBERS</td>
<td>KC2STA01</td>
<td></td>
</tr>
</tbody>
</table>

3. Correct any errors and re-run any jobs that did not complete successfully. Resubmit the KCIJcIVP job whenever the work output libraries change (for example, after applying maintenance and rerunning the $PARSE or $PARSEV job).

What to do next

After you successfully complete all the steps listed within the PARMGEN Workflow User Interface, a new runtime environment is created. Several steps must be completed outside of the PARMGEN Workflow User Interface before the runtime environment can be started. For more information, see Chapter 6, "Completing the configuration," on page 103.
Chapter 5. Configuring products with the Configuration Tool (ICAT)

This chapter describes the procedures for using the Configuration Tool for configuring the Tivoli Enterprise Monitoring Server on z/OS and z/OS-based monitoring agents:

If you are using the PARMGEN configuration method, skip this chapter and follow the instructions in Chapter 4, “Configuring products using the PARMGEN Workflow user interface,” on page 55.

Prerequisites

Before you can begin the procedures in this chapter, you must complete the SMP/E installation of any monitoring agents that you want to configure and the following Tivoli Management Services components:

- Configuration Tool
- Tivoli Enterprise Monitoring Server
- TMS:Engine

To install the components, follow the instructions in the IBM Tivoli Management Services on z/OS: Program Directory for IBM Tivoli Management Services on z/OS. To install monitoring agents, consult the Program Directory for each product.

The Configuration Tool provides default values wherever possible. These default values are generally sufficient to complete the installation, but can be changed to meet the requirements of your enterprise. For information about the features of the Configuration Tool, see Appendix B, “Configuration Tool reference,” on page 173.

Step 1. Create or update the Configuration Tool work library

You can set up the Configuration Tool in an existing consolidated software inventory (CSI), a cloned copy of an existing CSI, or a new CSI.

- If you are using an existing CSI that contains the Configuration Tool version 310.12 or higher, you can skip this step and go to “Step 2. Start the Configuration Tool” on page 78. The Configuration Tool libraries are updated automatically.
- If you are using a cloned copy of an existing CSI that contains the Configuration Tool V310.12 or higher, go to “If you use a cloned copy of a CSI containing the Configuration Tool.”
- If you are using a new CSI, go to “If you use a new CSI” on page 78.

Setting up the Configuration Tool consists of copying the contents from a target library into the Configuration Tool work library.

If you use a cloned copy of a CSI containing the Configuration Tool

If you use a cloned copy of an existing CSI that contains the Configuration Tool, follow these steps to update the work library:

1. Use the following JCL to create a job to copy the contents of the &thilev.TKCIINST library to the &shilev.INSTLIB library:

```
//COPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=* 
//IN DD DSN=&thilev.TKCIINST,DISP=SHR
//OUT DD DSN=&shilev.INSTLIB,DISP=SHR
//SYSIN DD *
COPY O=OUT,I=((IN,R))
```
where \&thilev\ is the SMP/E target high-level qualifier and \&shilev\ is the installation high-level qualifier.

**Tip:** To receive notification of the results of a job, add this option to your job card:

```
NOTIFY=userid
```

2. Submit the job.

**If you use a new CSI**

If you are using a new CSI, follow these steps to create the work library:

1. Customize and submit the KCIJALOC job, which you can find in the \&thilev\ .TKCIINST library (where \&thilev\ is the high-level qualifier of the target libraries that were installed by SMP/E). Follow the instructions in the JCL.

   The KCIJALOC job allocates the following installation libraries, which are required by the Configuration Tool:
   - \&shilev\ .INSTDATA
   - \&shilev\ .INSTDATW
   - \&shilev\ .INSTJOBS
   - \&shilev\ .INSTLIB
   - \&shilev\ .INSTLIBW
   - \&shilev\ .INSTLOG
   - \&shilev\ .INSTQLCK
   - \&shilev\ .INSTSTAT

   (where \&shilev\ is the high-level qualifier of the installation libraries). The job also copies elements from the \&thilev\ .TKCIINST library to the \&shilev\ .INSTLIB library.

2. Recycle the ISPF session.

---

**Step 2. Start the Configuration Tool**

Follow these steps to start the Configuration Tool:

1. Log on to a TSO session on the target system.

   The target system is the logical partition (LPAR) where you are going to create and configure a runtime environment for the Tivoli Enterprise Monitoring Server and any monitoring agents you intend to install and configure on the same LPAR.

2. Enter `ISPF`.

3. From the Primary Option Menu, enter option 6 (Command).

4. Turn off the predefined function (PF) keys so that the Configuration Tool function keys are not truncated. To do this, enter `PFSHOW` on the command-line and repeat this command until the ISPF default function keys are no longer displayed.

5. Enter the following command:

   ```
   EX '"shilev.INSTLIB''
   ```

   where \&shilev\ is the high-level qualifier that you specified when you created the INSTLIB data set (see "Step 1. Create or update the Configuration Tool work library" on page 77).

**Tip:** You do not have to specify a member name in this command.

A panel similar to [Figure 27 on page 79](#) is displayed, or a job is displayed that you can use to update the Configuration Tool to the most recent version.
Configuration tool usage notes

- Do not use the ISPF feature for edit recovery. If the ISPF RECOVERY ON command is entered, edits produce a recovery error message.
- Avoid using a split screen, because the last line of information at the bottom of the screen might not be displayed, and you cannot scroll down and might miss important messages or steps.
- Enter panelid in the OPTION ===> line to see the panel ID in the upper left corner of each Configuration Tool panel.
- The online help for the Configuration Tool contains detailed information about using the Configuration Tool panels. To display help information, press the F1 key (Help) or enter HELP on the command-line.
- README files provided by the Configuration Tool give additional usage information. If you enter R on the Runtime Environments panel, the Configuration Tool displays a list of available README files for selection. You can also enter README ccc (where ccc is the 3-character code for the type of README file) at the command-line of any Configuration Tool panel. For example, README RTE is the command for information about runtime environments, and README BAT is the command for information about batch mode. To obtain information about new features that affect the configuration of all components and products installed in the CSI, enter README AAA.
- The content of some files displayed by the Configuration Tool is generated dynamically and contains information specific to your environment. Examples include some of the online help, the README information, and the list of tasks shown in Complete the Configuration.

Step 3. Set up the Configuration Tool work environment

If you have not already done so, set up the Configuration Tool work environment by specifying the location of the work libraries, customizing the JCL for batch jobs, setting initial default values for the configuration environment, and allocating the work libraries.

To set up the work environment, follow these steps:

1. Specify the JCL options.
a. From the Main Menu panel shown in Figure 27 on page 79, enter 1 (Set up work environment). The Set Up Work Environment panel is displayed as shown in Figure 28.

![Set up Work Environment panel](image)

b. Enter 1 (Specify JCL options) to specify values for generating batch configuration jobs and for creating the work data sets required by the Configuration Tool. The Specify JCL Options panel is displayed as shown in Figure 29.

![Specify JCL Options panel](image)

c. Modify the values on the Specify JCL Options panel as required by your enterprise.

**JCL REGION value**
Specify the required storage value, in either KB or MB, for the REGION parameter on the EXEC statement of batch configuration jobs. The syntax is `numberK` or `numberM`. The default value is `0M` (zero MB).

**Installation work data sets**
Specify the attributes for allocating the work data sets required by the Configuration Tool.
### SMS considerations

If you plan to allocate SMS-managed data sets, ensure that the following conditions are all true:

- SMS is active on the z/OS image.
- The high-level qualifier of the data sets is eligible for SMS-managed volumes.
- You specify a combination of Volser, Unit, Storclas, and Mgmtclas parameters that is valid at your site. Because SMS can be implemented in several different ways, the Configuration Tool does not attempt to validate these parameters. The data set allocation jobs use the values you enter.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Specify the unit name to be used for allocating the non-VSAM work data sets. This value is required if the data sets are not managed by SMS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volser</td>
<td>Specify the volume serial numbers to be used for allocating the work data sets. This value is required if the data sets are not managed by SMS.</td>
</tr>
<tr>
<td>Storclas</td>
<td>If the data sets are to be managed by SMS, specify the SMS storage class to be used for the allocation. If your site does not require the SMS STORCLAS parameter, you can leave this field blank.</td>
</tr>
<tr>
<td>Mgmtclas</td>
<td>If the data sets are to be managed by SMS, specify the SMS management class to be used for the allocation. If your site does not require the SMS MGMTCLAS parameter, you can leave this field blank.</td>
</tr>
<tr>
<td>PDSE</td>
<td>If the non-VSAM data sets are to be managed by SMS, you can specify Y to allocate PDSE data sets instead of PDS data sets. PDSE data sets do not require compression and are not limited by a predefined number of directory entries. The default is N. Even if you specify Y, most load module libraries (Rkanmod, RKANMODL, RKANMODR, and RKANMODU) are not allocated as PDSE data sets. The main exception is RKANMODP, a load module library used for the SMP/E CALLLIBS facility.</td>
</tr>
</tbody>
</table>

### Job statement

Supply the standard jobcard to be used for batch jobs generated by the Configuration Tool. Control variables can be used but require an additional ampersand (&) for correct resolution of the variable. For example, if you use the &SYSUID control variable with the NOTIFY parameter in your job statements, specify NOTIFY=&SYSUID on this panel, as shown. This value resolves to &SYSUID in the batch job generated by the Configuration Tool.

If you use batch mode processing (see [Part 3, “Configuration Tool (ICAT) scenarios,” on page 119](#)), this standard jobcard is also used in batch mode configuration jobs. When you create the CICATB batch job member in the &shilev.INSTJOBS data set, the Configuration Tool also creates the batch jobcard ISP table KCITPIG1 in the &shilev.INSTDATA data set. The contents of the KCITPIG1 table are used as the standard jobcard in Configuration Tool-generated batch configuration jobs. If you modify the jobcard on this panel, you must recreate the CICATB batch job to refresh the KCITPIG1 jobcard table before running CICATB.

d. Press Enter to save the values and return to the Set Up Work Environment panel.

2. Set up the configuration environment.
   a. On the Set Up Work Environment panel (shown in Figure 28 on page 80), enter 2 (Set up configuration environment).
The Set Up Configuration Environment panel is displayed, as shown in Figure 30. The values specified on this panel become the default values for the first runtime environment you configure.

**Figure 30. Set up Configuration Environment panel**

b. Specify the high-level qualifiers for the VSAM, non-VSAM, and SMP/E runtime libraries. The length of the high-level qualifier for the runtime libraries must be no longer than 26 characters.

If high-level qualifiers for the runtime libraries have already been specified, unlock them by selecting Unlock runtime high-level qualifiers from the Configuration Services and Utilities panel [Figure 51 on page 173]. See Appendix B, "Configuration Tool reference," on page 173 for more information. The high-level qualifiers specified here are used as defaults for the first runtime environment you configure. After that, the default values are those of the runtime environment most recently worked on in the Configuration Tool.

c. Review the storage specifications and make any changes required for your enterprise.

d. Press Enter to save the values and return to the Set Up Work Environment panel.

3. Allocate the Configuration Tool work libraries.

a. On the Set Up Work Environment panel (shown in Figure 28 on page 80), enter 4 (Allocate work libraries).

The JCL to allocate the work libraries is displayed.

b. Review the JCL and edit if necessary.

The job allocates the following work libraries with the &shilev high-level qualifier:

- **INSTDATA** Contains all installation-specific information.
- **INSTDATW** Contains installation-specific information that was loaded from the distribution media.
- **INSTJOBS** Contains all generated jobs.
- **INSTLIBW** Contains the latest release of the Configuration Tool found on the distribution media. The contents of the INSTLIBW data set are moved to &shilev.INSTLIB unless the latest version of the Configuration Tool is already in the INSTLIB data set.
- **INSTLOG** Contains the cumulative configuration batch job log. For each job run, the log includes the data set and member name of the job; and for each job step, the log lists the date and time of job completion, the JES job name and number, the step name, the procedure step name, the completion code, and the name of the program executed.
**INSTQLCK** Used by internal installer processes as a locking data set. Used to prevent concurrent updates to the same data set.

**INSTSTAT** Contains status-specific installation information.

The default space allocations for the work libraries are adequate for most environments. However, the INSTDATA and INSTJOBS libraries expand when runtime environments are added and products are configured in those runtime environments. Review the space allocations specified in the JCL and increase them if required.

c. Submit the job to allocate the work libraries.
   If you do not submit the job while you are running the Configuration Tool, the job is stored in the CI$UPG2 member in the INSTJOBS data set. You can submit the job later when you exit the Configuration Tool. Depending on the environment, some of the data set members might not contain data.

d. Press F3 repeatedly until you exit the Configuration Tool. (The Allocate Work Libraries job cannot run while you are in the Configuration Tool.)

e. Verify that you have a return code of 0 for the Allocate Work Libraries job.

### Step 4. Set up and build a runtime environment

Before you configure monitoring agents or monitoring servers on a z/OS system for the first time, you must set up the runtime environments and build in which you will configure them. Review "Decision 7: What types of runtime environments to set up" on page 35 before you begin setting up your runtime environments.

After you have set up a runtime environment, you can proceed to configure products in it by following the product-specific instructions in the configuration guide for each product.

#### Tips

- If you intend to create runtime environments that share a base runtime environment, create the base runtime environment first.
- Before you begin using product-specific configuration instructions, read "Step 5. Configure your products and components in the runtime environment" on page 95, which suggests best practices for product configuration.
- See Table 21 on page 182 for more information on when each runtime environment configuration option is required.

Follow this procedure to set up and build a new runtime environment.

1. Access the Runtime Environments (RTE) panel.
   The Runtime Environment panel is one of the primary recurring panels in all configuration operations. This panel lists the actions you can perform to create and manage runtime environments.
   To access the Runtime Environments (RTE) panel, complete the following steps:
   a. If the Configuration Tool is not already running, start it:
      ```sh
      EX '&shilev.INSTLIB'
      ```
   b. From the Configuration Tool Main Menu (shown in Figure 27 on page 79), enter 3 (Configure products).
   c. On the Configure Products panel (Figure 31 on page 84), enter 1 (Select product to configure).
The Product Selection Menu (Figure 32) lists the products available for configuration.

Important

If you want to configure both a monitoring server and monitoring agents in the runtime environment, select one of the monitoring agents, not Tivoli Management Services on z/OS. If you select a monitoring agent, you can configure a monitoring server and one or more monitoring agents in the same runtime environment. If you select Tivoli Management Services on z/OS, you can only configure a stand-alone monitoring server in its own runtime environment, without any monitoring agents. For instructions on configuring a stand-alone monitoring server, see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.

The Runtime Environments (RTEs) panel (Figure 33 on page 85) is displayed.

d. Enter S (Select) to the left of the name of one of the monitoring agents or Tivoli Management Services.
2. Add a runtime environment.

Complete these steps to define each runtime environment before you configure products.

a. On the Runtime Environments (RTEs) panel, type A (Add RTE) in the Action field beside the first (empty) row and type a name for your new runtime environment in the Name field.

The runtime environment name is a unique identifier of up to 8 characters. It is automatically used as the mid-level qualifier for full and sharing runtime environments. You can optionally specify a mid-level qualifier for base runtime environments.

Tips

- Use a variation of the runtime environment name for the JCL suffix (used as the suffix of the name of the member containing the JCL in the INSTJOBS data set). If you specify a runtime environment name no more than 4 characters long, you can specify the same name for the JCL suffix. This setup makes it easy to identify the runtime environment associated with each job generated by the Configuration Tool and stored in the INSTJOBS data set.

- If you plan to use system variables, the runtime environment name must correspond in whole or in part to the value of the symbolic specified in a later panel [Figure 36 on page 90]. When you use system variables, components inherit the system values of the host z/OS system. These system-specific values are loaded into temporary data sets that exist only for the duration of the component execution. See Appendix A, “Enabling system variable support,” on page 165.

- When you enter a C (Configure), B (Build), or L (Load) next to the name of a runtime environment that has a previous version of the Tivoli Enterprise Monitoring Server installed, the Configuration Tool prompts you to confirm that you want to upgrade to the newer version. A batch job completes the upgrade and retains all previously configured values. See IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Upgrade Guide.

b. Specify the type of runtime environment being created. If you intend to create any sharing environments, start by creating the base or full runtime environment that they will share, so that you can then add the sharing runtime environment later. The following types are valid:
**Full**
Allocates both private and base libraries. Use this type if you require only one runtime environment or if you require a runtime environment for a unique set of products.

**Base**
Allocates read-only base libraries and does not execute alone. Use this type only in conjunction with sharing runtime environments. Create the base runtime environment first.

**Tip:** You cannot configure products in base runtime environments because they contain shared, read-only libraries exclusively.

**Sharing**
Allocates private libraries only. This type can either share base libraries with a base or full runtime environment, or use target libraries that were installed by SMP/E for its base libraries. Define one sharing runtime environment for each z/OS image.

For more detailed information about the different types of runtime environments, see "Decision 7: What types of runtime environments to set up" on page 35.

c. (For sharing runtime environments only) Type the name of the base or full runtime environment from which the sharing runtime environment obtains its base library information. If target libraries that were installed by SMP/E are to be shared, type SMP.

d. Type a description for this runtime environment. The description can be any information that is useful for you and others at your site.

e. When you have specified all required values on the Runtime Environments (RTEs) panel, press Enter.
   - If you are adding a base runtime environment, the panel shown in Figure 34 is displayed.

```
KCIPRTAB ------------------ ADD RUNTIME ENVIRONMENT --------------------------
COMMAND ===> RTE: BASE1 Type: BASE Desc: Base runtime environment

Libraries High-level Qualifier Volser Unit Storclas Mgmtclas POSE
Non-VSAM rhilev volser unit N
VSAM rhilev volser

Mid-level qualifier => rte (Optional for a base RTE)
JCL suffix => ssss
SYSOUT class => X
Diagostic SYSOUT class => X
Load optimization => N (Y, N)
Remote RTE for transport => N (Y, N)
Runtime members analysis => Y (Y, N)
```

**Figure 34. Add Runtime Environment panel for a base runtime environment**

- If you are adding a full or sharing runtime environment, the panel shown in Figure 35 on page 87 is displayed.
The values on the Add Runtime Environment panels are used to allocate runtime libraries and provide configuration defaults.

f. Use the following information to complete the first Add Runtime Environment panel.

**High-level qualifier**
Specify the high-level qualifiers to be used for allocating the runtime data sets. These values are required. The names of the runtime data sets are generated by appending a suffix to the high-level qualifier.

**SMS considerations**
If you plan to allocate SMS-managed data sets for the runtime environment, ensure that the following conditions are all true:

- SMS is active on the z/OS image.
- The high-level qualifier that you specify is eligible for SMS-managed volumes.
- You specify a combination of VOLSER, UNIT, STORCLAS, and MGMTCLAS parameters that is valid at your site. Because SMS can be implemented in several different ways, the Configuration Tool does not attempt to validate these parameters. The data set allocation jobs use the values you enter.

**Volser**
Specify the volume serial numbers to be used for allocating the runtime data sets. This value is required if the runtime data sets are not managed by SMS.

**Unit**
Specify the unit name to be used for allocating the non-VSAM runtime data sets. This value is required if the runtime data sets are not managed by SMS.

**Storclas**
If the runtime data sets are to be managed by SMS, specify the SMS storage class to be used for the allocation. If your site does not require the SMS STORCLAS parameter, you can leave this field blank.

**Mgmtclas**
If the runtime data sets are to be managed by SMS, specify the SMS management class to be used for the allocation. If your site does not require the SMS MGMTCLAS parameter, you can leave this field blank.
PDSE  If the non-VSAM data sets are to be managed by SMS, you can specify Y to allocate PDSE data sets instead of PDS data sets. PDSE data sets do not require compression and are not limited by a predefined number of directory entries.

    The default is N. Even if you specify Y, most load module libraries (RKANMOD, RKANMODL, RKANMODR, and RKANMODU) are not allocated as PDSE data sets. The main exception is RKANMODP, a load module library used for the SMP/E CALLLIBS facility.

Mid-level qualifier

    (Optional, for base runtime environments only) Specify a mid-level qualifier for base runtime library allocations. This field is modifiable for base runtime environments only. For full and sharing runtime environments, the mid-level qualifier is set to the runtime environment name and cannot be changed.

JCL suffix

    Specify a unique JCL suffix, no more than 4 characters long, to identify the batch job members created in the &shilev.INSTJOBS data set by the Configuration Tool for this runtime environment. Check the INSTJOBS data set to ensure that the suffix is not already in use.

    If possible, specify the runtime environment name (or an abbreviated version of the runtime environment name) as the JCL suffix. This setup makes it easy to identify the runtime environment associated with each job generated by the Configuration Tool and stored in the INSTJOBS data set.

STC prefix

    For full and sharing runtime environments, specify a prefix to be used when generating started task procedures for products configured in the runtime environment. The default value of CANS is provided, but specifying your own prefix prevents confusion with jobs generated by other runtime environments on the same system.

SYSOUT class

    Specify the value of the SYSOUT class for non-diagnostic output DDNAMEs, such as RKPDPLOG, in generated JCL. This value replaces the previously hardcoded SYSOUT class value.

Load optimization

    Specify whether you want to optimize the loading of this runtime environment, when the runtime environment load action (L) is selected after maintenance is applied or products are reconfigured.

    If you specify Y, the load job has the following characteristics:
    • Copies only modified modules from target to runtime libraries.
    • Requires access to IBM's SuperC (ISRSUPC) utility.
    • Uses less DASD space.
    • Performs additional analysis, which uses more CPU processing and file I/O.

    If you specify N (the default), the load job has the following characteristics:
    • Copies all members from target to runtime libraries, whether or not they were modified.
    • Requires more DASD space.
    • Uses less CPU time.

    Tip: Even if you enable load optimization, the first load operation copies all members to the runtime data sets. Load optimization takes effect on the second and subsequent load operations.

Remote RTE for transport

    Specify Y if you want to create this runtime environment with the intention of transporting it to other systems, rather than of running it on the local system. When you finish adding the runtime environment, you skip building the runtime libraries (action B on Figure 33 on page
You configure the runtime environment (action C), but you do not run the generated configuration batch jobs or load the libraries (action L). Subsequently, you can transport the runtime environment to other systems, where you build the runtime libraries, run the configuration batch jobs, and load the runtime libraries. Because there are no runtime libraries on the system where the runtime environment was originally created, you cannot analyze the libraries for user-modified elements on that system. But you can analyze the libraries on the first system to which you transport the runtime environment and on which you build the libraries by using the Analyze user-modified elements option.

The default is N.


Runtime members analysis
The Configuration Tool generates configuration batch jobs for the runtime environment and can identify user-modified data set members that each batch job will affect. If runtime members analysis is enabled, a report of user-modified members is displayed. You can also generate these reports from the RTE Utility Menu (action Z).

The default is Y.

Diagnostic SYSOUT class
Specify the value of the SYSOUT class for diagnostic output DDNAMEs, such as SYSUDUMP and SYSABEND, in generated JCL. This value replaces the previously hardcoded SYSOUT class value. The default is X.

Will this RTE have a Tivoli Enterprise Monitoring Server?
This question is asked for full and sharing runtime environments only. If you are creating a runtime environment for stand-alone monitoring agents without a monitoring server, specify N. Otherwise, specify Y (the default) to allocate libraries for the monitoring server.

Be sure to make a note of the TEMS name value displayed here. You must specify it when you configure components and products in the runtime environment, and when you set up communications with the distributed components of Tivoli Management Services (IBM Tivoli Monitoring). The TEMS name parameter is case sensitive on all platforms.

Copy configuration values from RTE
For full and sharing runtime environments, you can specify the name of an existing runtime environment from which you want configuration values to be copied for this runtime environment. This procedure makes an exact copy of the existing runtime environment. If the two runtime environments will not contain the same products, do not use this procedure.

After you have specified all required values on the first Add Runtime Environment panel, press Enter.

• If you are adding a base runtime environment, the Runtime Environments (RTEs) panel (Figure 33 on page 85) is displayed. This step completes the addition of the runtime environment. You must build the runtime libraries before adding a sharing runtime environment for the product-specific libraries. Go on to 3 on page 95.

• If you are adding a full or sharing runtime environment, the second Add Runtime Environment panel (Figure 36 on page 90) is displayed.
h. Use the following information to complete the second Add Runtime Environment panel.

**Use z/OS system variables?**

Specify Y if you want to use z/OS system variables. The default is N.

When you use system variables, components inherit the system values of the host z/OS system. These system-specific values are loaded into temporary data sets that exist only for the duration of the component execution. See Appendix A, “Enabling system variable support,” on page 165.

**Tips**

- If system variable support is enabled, you can enter README SYS at the command-line for more information on how the Configuration Tool uses z/OS system symbols to process VTAM applids.
- For complete information about z/OS system symbols, see the z/OS MVS Initialization and Tuning Reference.

If you specify Y for system variables, supply values for the remaining fields in this section:

**RTE name specification**

Specify the system variable for the name of the runtime environment. The default is &SYSNAME.

The resolved name must be a single valid JCL symbol, as defined in the z/OS MVS JCL Reference. This field becomes the value of the SYS= parameter (for example, SYS='&SYSNAME') in all started task members. This field can contain both literals and symbolics. For example, if you specify a value of CAN&SYSNAME, the value resolves to a runtime environment name of CANSYSA if &SYSNAME=SYSA.

**RTE base alias specification**

If the runtime environment is sharing with a base runtime environment, specify a system variable for the base runtime environment. This value is then inserted into the base runtime environment data set references in all started tasks. The resolved name must be a valid data set name qualifier.

If the runtime environment you are defining is not sharing with a base runtime environment, the label n/a is displayed beside this field.
Applid prefix specification
If you are using system variables, enter a VTAM applid prefix that contains system variables. The resolved prefix can be a maximum of 4 characters. Be sure to place a period (dot) after the last symbolic in the specification. &SYSCLONE. is the default.

For information on system variable considerations for the VTAM Major Node name, enter README SYS at the command-line and read the sample usage scenario.

Use VTAM model applids?
Specify Y if you want this runtime environment to use VTAM model applids; that is, applids with wildcard suffixes such as ? or *. These model applids allow the use of any applids that match the pattern in the VTAM node.

The default is N.

Security system
Specify which, if any, security system is to be used for this runtime environment. The default is NONE. If you specify a security system, verify that it is installed and configured correctly for your site. If you specify ACF2, you must also provide the name of the ACF2 macro library.

Tip: Specifying a security system here indicates which system will be used for security validation of users signing on to the Tivoli Enterprise Portal, but it does not enable the validation. Security validation of users is enabled in a Tivoli Enterprise Monitoring Server configuration panel. For more information about configuring security, see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.

Fold password to upper case
By default, TMS:Engine converts logon passwords to uppercase. However, RACF V1.7 supports mixed-case passwords. If you want to implement mixed-case passwords and if all your monitoring agents support them (this requires application of maintenance for each monitoring agent), set this field 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. Leave this field with the default value Y.

For more information about the MIXEDCASE option in RACF, see the z/OS V1R8.0 Security Server RACF Security Administrator's Guide. For more information about the way TMS:Engine handles security, enter README SEC at the command-line of any Configuration Tool panel.

Global SAF class name
Specify a common System Authorization Facility (SAF) security class name for OMEGAMON enhanced 3270 user interface security controls. The value is used in the RTE_SECURITY_CLASS parameter in the KppENV member of the RKANPARU library. If this field is not specified, the RTE_SECURITY_CLASS parameter is generated as a placeholder comment in the RKANPARU(KppENV) of components exploiting the parameter. Specify a valid SAF class name if you want to specify a value. If you are using ACF2 as your external security resource manager, specify a maximum of three characters.

VTAM communication values
VTAM communication values are required for some monitoring agents and for the monitoring servers with which those monitoring agents communicate directly. Examples of such monitoring agents include OMEGAMON XE on z/OS (for the EPILOG facility of the OMEGAMON II component) and OMEGAMON XE for Messaging on z/OS (for the 3270 interface component). See the product-specific configuration guides for further information about SNA requirements.
The values you specify here become the default values for the products and components configured in the runtime environment. If none of the products or components are to use the SNA communication protocol (for example, if you are defining a runtime environment for a high-availability hub), clear these fields.

**Applid prefix**
Type a global VTAM applid prefix (from 1 to 4 characters) to be used when building the VTAM applids for products in this runtime environment. The default is CTD. If none of the products or components you intend to configure in this runtime environment require SNA communications, clear this field.

**Tips**
- If you have enabled system variables, the applid prefix value specified in the system variables section of this panel is used instead of the value specified in the VTAM communication values section.
- Enter README APP on the command-line to review information about default applids and understand how the Configuration Tool processes VTAM applids.

**Network ID**
Identify your VTAM network, as defined in the NETID parameter of the VTAMLST startup member ATCSTRnn. This parameter is required for SNA communications. If none of the products or components you intend to configure in this runtime environment require SNA communications, clear this field.

**Logmode table**
Type the logmode table name for LU6.2 logmode entries. The default is KDSMTAB1. If none of the products or components you intend to configure in this runtime environment require SNA communications, clear this field.

This table is assembled into the system library (usually SYS1.VTAMLIB) that contains VTAM logmode tables.

**LU6.2 logmode**
Type the LU6.2 logmode for this runtime environment. The default is CANCTDCS. If none of the products or components you intend to configure in this runtime environment require SNA communications, clear this field.

**TCP/IP communication values**
The values you specify here become the default TCP/IP values for the products and components configured in the runtime environment.

**Tip:** In the TCP/IP communication values section of the panel, the fields Hostname, Address, and Started task support system variables defined in the system variable parameter member, but the fields require literal values for input on this panel. Specify the literal values here, and specify the symbolic variables in the system variable parameter member. For more information, issue the README SYS command from any Configuration Tool panel.

**Hostname**
The TCP/IP host name or dotted-decimal IP address of the z/OS system where the runtime environment is being defined. To obtain the host name and IP address, enter TSO HOME TEST at the command-line.

If the z/OS domain name resolver configuration specifies a search path that includes the target domain suffix, specify only the first qualifier of the host name. (Example: sys is the first qualifier of the fully qualified host name sys.ibm.com.) Otherwise, specify the fully qualified host name.
Started task
Identifies the TCP/IP stack to be used. If the LPAR contains a single TCP/IP stack, accept the default value, which is the wildcard asterisk character (*), to select the first TCP/IP stack that was started.

If the LPAR contains more than one TCP/IP stack, you can specify the started task name of the TCP/IP stack you want to use; or you can specify the number 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.

Tip: 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.

If IP domain name resolution is not fully configured on the z/OS system, the SYSTCPD DD statement is required (see “Add support for the SYSTCPD DDNAME in the started tasks” on page 103).

Port number
The number of the well-known port to be used for IP communications. The default is 1918. (See “Port number assignments” on page 31.)

i. When you have finished defining the runtime environment, press Enter. A third Add Runtime Environment panel is displayed (Figure 37).

--- ADD RUNTIME ENVIRONMENT (3 of 3) ---
COMMAND ===>

If you require TN3270E Telnet session link support override, complete these values:
Hostname ==> 
Port number ==> 
LUGROUP ==> 

If any products in this RTE require Unix System Services (USS) directories be created, specify the RTE HFS/zFS home directory:
RTE home directory ==> /rtehome______________________________ (rtedir)
(Press F1=Help for rtedir RTE path name considerations.)

Enter=Next F1=Help F3=Back

Figure 37. Add Runtime Environment (3 of 3) panel for a full or sharing runtime environment

j. Complete this panel if at least one of the products to be configured in this runtime environment has at least one of the following conditions:
• The product supports the Dynamic XE to 3270 (Classic) linking feature, and you must override the default values.
  The default values or the override values you specify on this panel are displayed during TN3270 logon and can be modified then for an individual TN3270 session.
• The product requires UNIX System Services (USS) directories to be created (for example, if the 
agent stores files in the USS file system).

Use the following information to complete this panel:

Hostname
If the LPAR in which you are defining a runtime environment does not have an active Telnet 
listener, specify the network address of an LPAR that does have an active Telnet listener. A 
network address can be specified as one of the following values:
• Fully qualified hostname (for example, sys.ibm.com)
• First qualifier of the fully qualified hostname (for example, sys for sys.ibm.com)
• 32-bit IPv4 address in dotted decimal notation (for example, 9.67.1.100)

To get this value, issue the TSO HOMETEST command on the LPAR of the Telnet listener.

Port number
The default port number of the Telnet listener is 23. To override this value, specify the port 
number of the Telnet listener.

LUGROUP
The Dynamic XE to 3270 (Classic) linking feature requires the VTAM Unformatted System 
Services screen to accept a LOGON APPLID() DATA() command. If the default Telnet USS 
screen does not accept this command, supply the name of a Logical Unit (LU) group that 
does accept it. The TN3270 session will be joined to that LU group.

RTE home directory
If a product to be configured in the runtime environment requires USS directories to be 
created, specify the main Hierarchical File System/zSeries File System (HFS/zFS) directory. 
The directory name must begin with a slash (/) and represents the #rtedir value (the name 
of the file system directory where the runtime environment is to be created). Examples:
/omonrte/rte1/kan/bin
/omonrte/etc/rte1/kan/bin
/omonrte/rte1/kds/support/TEMS

in the following format:
#rtedir/#rtename/#ppp/#filetype
#rtedir/#rtename/#ppp/#workarea (/#filetype)

where:
#rtedir = /omonrte or /omonrte/etc
#rtename = rte1
#ppp = kan or kds
#filetype = bin
#workarea = /support/TEMS

Security tip
Some products create their own USS directories with a default group and user USS 
directory permissions of read/write/execute (777). Other products require a higher level 
of permissions for the USS directories and subdirectories they use. To implement a 
more secure access scheme for the USS directories of the runtime environment, use 
group-based security. For detailed information, enter README USS on the command-line.

k. Press Enter to return to the Runtime Environments (RTEs) panel Figure 33 on page 85.
3. Build the runtime libraries. This step generates JCL to allocate the required runtime libraries for the selected product. You must perform this step for all runtime environments you create, including base runtime environments. You must also perform this step when you are installing a new product into an existing runtime environment. Note that if you are setting up a base runtime environment, only shared libraries are allocated, not product-specific libraries.

The libraries built by this step depend on the product selection made on the Product Selection Menu. If you want to add other products to the runtime environment, you must select each from the Product Selection Menu and perform the Build option for each runtime environment in which you want to configure it.

a. On the Runtime Environments (RTEs) panel, type B to the left of the name of the runtime environment for which you want to build the libraries, and press Enter.

   The JCL to allocate the runtime libraries is displayed.

b. Review the JCL, edit if necessary, and submit the job.

   You might want to change the jobname to match the member name so that you can identify the jobs later on.

c. Verify that the job completes successfully.

   Expect to receive a return code of zero. If you do not, check the log to diagnose errors, and then rerun the job.

d. Press F3 (Back) to return to the Runtime Environments (RTEs) panel.

   If you have built the runtime libraries for a base runtime environment, you must add a sharing runtime environment for product-specific libraries. Go back to page 85 and complete the steps for defining a sharing runtime environment and building its libraries.

   Note that placing B beside a sharing-with-base or a sharing-with-full runtime environment builds the libraries for both the shared runtime environment (base or full) and the sharing runtime environment.

### Step 5. Configure your products and components in the runtime environment

This step describes best practices that apply to the configuration of all products. Read those topics before you start configuring your products according to the instructions in the individual product configuration guides.

#### Configure a monitoring server

The product configuration menu for every OMEGAMON XE monitoring agent on z/OS and for Tivoli Management Services on z/OS begins with an option to configure a monitoring server. IBM Tivoli OMEGAMON XE on z/OS and IBM Tivoli OMEGAMON XE for Storage on z/OS require that the agents run in the same address space as a monitoring server, so you must configure a monitoring server first in every runtime environment in which you configure an OMEGAMON XE on z/OS or OMEGAMON XE for Storage monitoring agent.

Monitoring agents that can be configured to run in their own address space (stand-alone) do not require that a monitoring server be configured in the same runtime environment.

If you are configuring a monitoring server in the runtime environment, configure it before you configure any monitoring agents. Do not configure more than one monitoring server in an LPAR, unless you are
configuring a high-availability hub in one runtime environment and a remote monitoring server and agents in another runtime environment on the same LPAR. For instructions on configuring a high-availability hub, see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.

Set appropriate parameter values
In most cases, you set parameter values for a runtime environment and for the products and components in the runtime environment by accepting defaults or supplying values in the Configuration Tool interactive panels, and then you use batch mode and the batch parameter member to propagate the parameter values to other LPARs. (See Part 3, “Configuration Tool (ICAT) scenarios,” on page 119.) However, in special cases, under the guidance of IBM Software Support, you might have to add parameters or parameter options that are not presented in the Configuration Tool interactive panels.

This section explains how to use the Configuration Tool to obtain parameter reports, modify parameter values in already-configured runtime environments, and add parameters to members of your runtime libraries:

Obtaining parameter reports
You generate parameter reports, also called parameter maps, from option 11 (Verify Configuration and Generate Parameter Map) on the RTE Utility Menu (Figure 52 on page 175). When you select this option, the Verify Configuration and Generate Parameter Map panel is displayed (Figure 38).

The type of parameter map you can generate depends on whether you have already configured components and products in the selected runtime environment.

- To generate a parameter map after creating a runtime environment but before configuring any components or products in it, select option 2 (Generate parameter map).
  The parameter map lists all parameters associated with the components and products that were installed into the target libraries by SMP/E, and gives a short description and other basic information about each parameter.

- To generate a parameter map after configuring components and products in the runtime environment, follow these steps:
  1. Select option 1 (Verify configuration) to run a batch job that produces a configuration verification report for the runtime environment.
     The Configuration Tool displays the JCL for the &shilev.INSTJOBS(IVP@ssss) batch job, where ssss is the JCL suffix for the runtime environment.
  2. Submit the configuration verification batch job.
     The configuration verification job generates a report that lists the required data sets and members; indicates the batch job that creates each one; identifies which data sets and members are missing, and which batch jobs are missing or did not complete successfully; and specifies which of the
missing or unsuccessful batch jobs and which of the missing data sets and members are required and which are optional. If a large number of components and products are configured in the runtime environment, the configuration verification batch job might take a long time to run.

3. Check to make sure that the configuration verification batch job has run successfully and that the configuration verification report is stored in the &shilev.INSTJOBS.IVP$ssss(IVP1ssss) member.

4. Select option 2 (Generate parameter map) on the Verify Configuration and Generate Parameter Map panel.

   The Configuration Tool generates a parameter map listing all parameters for the configured products and components in the runtime environment, along with the value for each configured parameter and other post-configuration parameter information such as the name of the configuration job that stored each parameter and data about completion of each job and each step. The job and step completion codes reported in the parameter map are retrieved from the configuration batch job log stored in the &shilev.INSTLOG data set.

   The parameter map is generated in the foreground and is displayed as soon as it is complete. If a large number of components and products are configured in the runtime environment, generating the parameter map might take several minutes.

   The parameter map for a runtime environment is stored in the &shilev.INSTJOBS.IVP$ssss(PMAPssss) member.

The top of each parameter map provides the following information to help you locate key data sets and files for a configured runtime environment:

RTE Name: rte_name
JCL Suffix: ss
INSTJOBS: &shilev.INSTJOBS
INSTLOG: &shilev.INSTLOG
IVP Detail: &shilev.INSTJOBS.IVP$rte_name(IVPDssss)
INSTDATA: &shilev.INSTDATA
TKANCUS: &shilev.TKANCUS

See the following example of the kinds of information the parameter map provides for each configured parameter:

NAME: KDS_CMS_STC
Description: 'Server started task name'
VARNAME: DSSPPMEM
VALUE: CIDSST
PANELS: 1 PANEL DESCRIPTION
-------- -----------------------------------------------------
KDS62PP1 Specify Configuration Values

SKELS: 1 KDS621SSC
MEMBERS: 1 MEMBER JOB STEP LIB JOBNAME JOB# HI-CC
-------- -------- -------- -------- -------- ------ -----
CIDSST DS#3ssss STEP1 RKANSAMU userid$ST J19881 00000
CIDSST DS#4ssss STEP1 RKANSAMU userid$ST J19916 00000
CIDSST MQ#4ssss STEP1 RKANSAMU userid$ST J19943 00000

Name Batch parameter name
Description Brief text describing the parameter
VARNAME For IBM internal use only
Panel Panel ID of a Configuration Tool interactive panel on which the parameter value is set
Description Panel name of a Configuration Tool interactive panel on which the parameter value is set
SKELS For IBM internal use only
Member
Name of a runtime member where the parameter value is stored

Job
Name of a batch job that generates the parameter value from Configuration Tool input

Step
Name of a job step that generates the parameter value from Configuration Tool input

Lib
Runtime library where the parameter value is stored

Jobname
ID of the job card submitter

Job#
Number of the submitted job

HI-CC
Highest condition code resulting from submission of the job

Tips

- A parameter map is specific to one runtime environment. Generate a separate parameter map for each runtime environment.
- Only one parameter map can be stored for each runtime environment. If you want to keep a previously generated parameter map (for example, if you want to keep the pre-configuration map when you generate a post-configuration map), back it up before you generate a new parameter map for the runtime environment.
- If you subsequently reconfigure components or products, or configure additional ones, in the runtime environment, regenerate and rerun the configuration verification job and then regenerate the parameter map.
- For detailed information about the configuration verification report and parameter map, enter README IVP at the command prompt from any Configuration Tool panel.

Modifying configured parameter values

After you finish configuring your products, you might have to change the value of one or more parameters. For example, it is a best practice to disable security validation on the hub monitoring server until after you finish configuring all components and verifying that you can start them and that they can communicate with each other. At that point, you probably want to enable security validation on the hub. You do this by changing the value of a parameter.

It is possible to change a parameter value by editing the runtime member where the value is stored. However, if you edit a runtime member directly, your changes are overwritten when you reinstall or reconfigure the product. Therefore, it is best to use the interactive panels of the Configuration Tool to change parameter values that were originally set in the Configuration Tool. If you want to make the same change on every LPAR, you can then use batch mode to propagate the change to your other runtime environments.

Adding, changing, or deleting a parameter in a runtime member

In special cases, under the guidance of IBM Software Support, you might have to add, change, or delete parameters or parameter options that are not presented in the Configuration Tool interactive panels. For example, “Decision 6: How to set up communications between components” on page 27 explains changes to the KDE_TRANSPORT environment variable that might be required in certain environments. Some KDE_TRANSPORT options, such as the selection of communication protocols to be used by monitoring servers and monitoring agents, can be set in the fields of the Specify Communication Protocols panel (for a monitoring server) or the Specify Agent Address Space Parameters panel (for a stand-alone monitoring agent). However, some KDE_TRANSPORT options do not have fields in those panels.

Parameters or parameter options that do not correspond to fields in the Configuration Tool interactive panels are known as nonstandard parameters. To add, modify, or delete a nonstandard parameter, use the Specify Nonstandard Parameters panel.
Important
Use extreme caution in specifying nonstandard parameters. No error checking is provided and syntax is not validated. The presence of the data set and member specified is not validated. If faulty syntax or other errors cause your edits to fail, no warning or error message is issued. Use this facility only under the guidance of IBM Software Support.

To display the Specify Nonstandard Parameters panel (shown in Figure 39), press F5=Advanced from the Specify Advanced Configuration Values panel for a monitoring server or the Specify Advanced Agent Configuration Values panel for a monitoring agent.

Use the following information to complete the Specify Nonstandard Parameters panel:

Parameter
The name of the configuration parameter to be added, replaced, or deleted.

New Value
The value you want the parameter to have. Include format characters. (For example, if the format in the runtime member is parameter=value, precede the value with an equal sign; or if the format is parameter(value), surround the value with parentheses.) If you want to delete an existing parameter, leave the New Value field blank.

Old Value (if replacing)
The existing parameter value that you want to replace or delete. Include format characters. The character string in the Old Value field must match exactly the existing value for the parameter in the runtime member, or the value is not replaced.

Low-level data set qualifier
The low-level qualifier of the data set containing the member with the parameter to be added, replaced, or deleted.

Member
The member containing the parameter to be added, replaced, or deleted.
For each runtime environment, you can specify a maximum of 18 parameters for each monitoring server address space, a maximum of 6 parameters for each stand-alone OMEGAMON XE monitoring agent address space, and a maximum of 12 parameters for each OMEGAMON (3270) or OMEGAMON II components.

**Examples**

---

**Adding a parameter:**

<table>
<thead>
<tr>
<th>Parameter:</th>
<th>PARM1 __________________________</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Value:</td>
<td>=YES ___________________________</td>
</tr>
<tr>
<td>Old Value (if replacing):</td>
<td>____________________________________</td>
</tr>
<tr>
<td>Low-level data set qualifier:</td>
<td>RKANPARU Member: KDENV__</td>
</tr>
</tbody>
</table>

---

**Changing the value of an existing parameter:**

<table>
<thead>
<tr>
<th>Parameter:</th>
<th>PARM1 __________________________</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Value:</td>
<td>=NO ___________________________</td>
</tr>
<tr>
<td>Old Value (if replacing):</td>
<td>=YES ___________________________</td>
</tr>
<tr>
<td>Low-level data set qualifier:</td>
<td>RKANPARU Member: KDENV__</td>
</tr>
</tbody>
</table>

---

**Deleting a parameter:**

<table>
<thead>
<tr>
<th>Parameter:</th>
<th>PARM1 __________________________</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Value:</td>
<td>__________________________________</td>
</tr>
<tr>
<td>Old Value (if replacing):</td>
<td>=YES ___________________________</td>
</tr>
<tr>
<td>Low-level data set qualifier:</td>
<td>RKANPARU Member: KDENV__</td>
</tr>
</tbody>
</table>

---


**Verify the configuration before loading the runtime libraries**

By using the **Verify configuration** option of the RTE Utility Menu (Figure 52 on page 175), you can verify the creation and successful execution of all configuration batch jobs, and the creation of all required runtime data sets and members. You can run the configuration verification job by selecting this option at any time; it is a good idea to run it as soon as you finish configuring a runtime environment and before you load the runtime libraries. Additionally, the configuration verification job is run automatically when you load the runtime libraries. This job can be run in batch mode only; it cannot be run in foreground. If a large number of components and products are configured in the runtime environment, the configuration verification batch job might take a long time to run.

After you run the configuration verification job, you can also generate a detailed parameter map for your newly configured runtime environment. The detailed parameter map describes each parameter, shows the value for each configured parameter, and provides other post-configuration parameter information such as the name of the configuration job that stored each parameter and data about completion of each job and each step.

Enter README IVP on the command-line for more information about the configuration verification report and the parameter map. You can also find more information in "Runtime environment utilities" on page 174 and "Obtaining parameter reports" on page 96.

**Load runtime libraries once for multiple products**

The last step in configuring each product using the Configuration Tool is loading its runtime libraries. Each time you run the Load function, the Configuration Tool loads the runtime libraries for the entire runtime.
environment. If you are configuring multiple products in the runtime environment at one time, you can wait until you have configured all the products and then perform the Load step. For more information, see "The Runtime Environments (RTE) panel" on page 180.

However, if different people are configuring different products, or if you are configuring products in the runtime environment at different times, load the runtime libraries after you configure each product.

Eliminate duplicate Complete the Configuration steps

At the end of each product configuration menu is a step to display a list of Complete the Configuration tasks. You do not have to perform these tasks immediately after you finish configuring each product. If you are configuring multiple products within a runtime environment, you can postpone these additional tasks and perform them after you finish configuring all the products that you want to configure at this time. You can use the Configuration Tool to generate a concatenated list of those tasks for all the products in a runtime environment. For information about the runtime environment utilities in the Configuration Tool, see "Runtime environment utilities" on page 174.

A number of steps are listed in the Complete the Configuration list for every product. These repeated steps can be performed once for all the products in a runtime environment:

- "Add support for the SYSTCPD DDNAME in the started tasks" on page 103.
- "Copy started task procedures to your procedure library" on page 103.
- "Copy the VTAM definitions to your system VTAMLST" on page 104.
- "Vary the VTAM major node active" on page 105.
- "APF-authorize the runtime load libraries" on page 105.
- "Complete the configuration for the OMEGAMON subsystem (optional)" on page 106.
- "Complete the configuration for OMEGAMON monitoring agents" on page 107.
- "Enable the self-describing agent feature or manually install application support" on page 107.
- "Install language support" on page 108.
- "Verify the configuration" on page 108.
- "Enable security" on page 111.
- "Enable security on the IBM Tivoli OMEGAMON enhanced 3270 user interface" on page 112.
- "Enable historical data collection (optional)" on page 116.
- "Run the ITMSUPER Tools (optional)" on page 117.

If you are configuring several products in a runtime environment at one time, or if you are using batch processing to replicate existing runtime environments, you can use the steps in the following scenario:

1. Configure your products in the runtime environment, or configure the runtime environment by using the CICATB batch job. Until you have finished configuring all products in the Configuration Tool, postpone loading the runtime libraries and performing any Complete the Configuration steps outside the Configuration Tool for any products.
2. Run the runtime environment Load job (Load All Product Libraries After SMP/E). If you use CICATB batch job, this step is completed for you.
3. Generate the concatenated list of Complete the Configuration steps for the runtime environment. For more information, see "Runtime environment utilities" on page 174.
4. Review and perform the Complete the Configuration steps.
5. Review and perform the product-specific Complete the Configuration steps for each product configured in the runtime environment. Skip the common steps that you have already performed.

If different people are configuring different products, or if you are configuring products at different times, perform the Load and Complete the Configuration steps for each product as you configure it to avoid missing required steps. Chapter 6, “Completing the configuration,” on page 103 describes the actions required to complete each of the required steps and the recommended order in which to complete them.
Chapter 6. Completing the configuration

After you finish the configuration steps described in either Chapter 4, "Configuring products using the PARMGEN Workflow user interface," on page 55 or Chapter 5, "Configuring products with the Configuration Tool (ICAT)," on page 77, you must take additional steps to complete the configuration of your runtime environment. The steps that you are required to complete depend on the steps that you have already taken, the configuration options that you have chosen, and the components or products that you are configuring. Some of the steps are required and others are optional.

Many of the steps required are the same for multiple products and can be completed all at one time for all the products. To help you configure your components correctly, work through this chapter in conjunction with the corresponding chapter within the product-specific Planning and Configuration guides.

The rest of this chapter gives basic instructions for completing the configuration of common or shared components within your runtime environment. On completion, be able to verify that all components have started correctly and be able to complete the configuration of optional advanced features such as collecting historical data and enabling security.

Add support for the SYSTCPD DDNAME in the started tasks

Note: This step is applicable only if you configured your runtime environment by using the Configuration Tool (ICAT).

SYSTCPD explicitly identifies which data set to use to obtain the parameters defined by TCPIP.DATA when no GLOBALTCPIPDATA statement is configured. If a monitoring server is using any of the IP.UDP-related or IP.PIPE-related communication protocols for connection, but the IP domain name resolution is not fully configured on the z/OS system, the SYSTCPD statement must be supported by the monitoring server and the monitoring agents that report to it.

If you are certain that SYSTCPD is not required in your environment, you can skip this step. However, note that you might gain a small performance benefit by avoiding multiple dynamic data set allocations if you supply a SYSTCPD DD statement.

To support SYSTCPD, uncomment the following statement in the started task members in RKANSAMU and provide the name of the SYSTCPD data set:

```plaintext
//*SYSTCPD DD DISP=SHR,
//* DSN=TCPIP.SEZAINST(TCPDATA)
```

The name of the SYSTCPD data set is installation-specific. Get the correct specification from your network administrator.

If you reconfigure products, you must uncomment the statement again and refresh the monitoring server and monitoring agent started tasks from RKANSAMU to PROCLIB.

Copy started task procedures to your procedure library

Note: This step is applicable only if you configured your runtime environment by using the Configuration Tool (ICAT).

During configuration, several started task procedures are created in the &rhilev.&rte.RKANSAMU data set (and in the &rhilev.&rte.RKD2SAM data set if the IBM Tivoli OMEGAMON XE for DB2 Performance
Expert on z/OS or IBM Tivoli OMEGAMON XE for DB2 Performance Monitor on z/OS product or both are being configured). The following procedures must be copied to your procedure library under a user ID with Write authority to the PROCLIB data set:

- CANSDSST (the monitoring server started task).
- KPDPROC1 (maintenance procedure for the persistent data store). The KPDPROC1 procedure is present only if you configured a persistent data store in the runtime environment.
- Started task procedures for any monitoring agents configured in their own address spaces in the runtime environment.

You can copy the procedures manually or use the runtime environment sample system procedure copy JCL within the Configuration Tool. To copy the procedures by using the utility, perform the following steps:

1. On the Runtime Environments (RTE) panel, type Z (Utilities) next to the runtime environment whose started tasks you want to copy and press Enter.
2. From the RTE Utility Menu, enter 6 (Generate sample system procedure copy JCL).
3. On the next panel, type the name of your procedure library, for example, SYS1.PROCLIB, and press Enter.
   The JCL (CB#PIDD) that creates the copy job is displayed.
4. Edit the JCL and submit the job. Expect a return code of zero.
   This step generates a member called KCISYPJB in the &rhilev.&rte.RKANSAMU and RKD2SAM libraries. This member lists the started tasks for all products configured in the runtime environment.
5. Edit the JCL if necessary and submit KCISYPJB from a user ID that has authority to the PROCLIB data set.
   This job copies all the required started tasks to the specified procedure library. Expect a return code of zero.

Be careful not to overwrite any members in the data set that have already been modified. If necessary, copy the started task procedures manually.

---

**Copy the VTAM definitions to your system VTAMLST**

**Note:** This step is applicable only if you configured your runtime environment by using the Configuration Tool (ICAT).

During configuration, the Configuration Tool also creates VTAM definitions in the &rhilev.&rte.RKANSAMU data set (and in the &rhilev.&rte.RKD2SAM data set if the IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS or IBM Tivoli OMEGAMON XE for DB2 Performance Monitor on z/OS product or both are being configured). Under a user ID with write authority to the VTAMLST data set, copy the VTAM major nodes to your system VTAMLST. You can copy the definitions manually or use the Configuration Tool utility to copy them.

To copy the definitions by using the utility:

1. On the Runtime Environments (RTEs) panel, type Z next to the name of the runtime environment whose nodes you want to copy and press Enter.
2. From the RTE Utility Menu, select option 7 (Generate sample system VTAMLST copy JCL) and then press Enter.
3. On the next panel, type the name of your VTAM node, and press Enter.
   The JCL to create the KCISYNJB member is displayed.
4. Edit the JCL and submit the job. Expect a return code of zero.
   The KCISYNJB member is created in the &rhilev.&rte.RKANSAMU and RKD2SAM libraries.
5. Edit the KCISYNJB member and submit the job from a user ID that has authority to the VTAMLST data set.
This job copies all the required VTAM definitions to the specified VTAMLST. Expect a return code of zero.

Be careful not to overwrite any members in the data set that have already been modified. If necessary, copy the definitions manually.

**Tip**

You can create a single VTAM major node that contains applids for all the products configured in the runtime environment. Creating a single node can simplify copying and activation. To create a single major node:

1. On the Runtime Environments (RTEs) panel, type 2 next to the name of the runtime environment whose nodes you want to copy and press Enter.
2. On the RTE Utility Menu, type 4 (Create VTAM major node) and press Enter.
3. On the Create VTAM Major Node panel, type a name for the node and press Enter.
   - The JCL for the job (CB8ssss) that creates the single major node is displayed
4. Review the JCL, and then submit the job to create the node.

This procedure must be repeated if the runtime environment values change. If a new product is added to the runtime environment or an existing product is reconfigured to change any of the system variable applid values, rerun the CB8ssss job.

**Vary the VTAM major node active**

**Note:** This step is applicable only if you configured your runtime environment by using the Configuration Tool (ICAT).

Issue the following command to vary a VTAM major node active:

```
V NET,ACT,ID=nodeid
```

If you have not created a single VTAM major node for all the products, repeat the command for each node.

**APF-authorize the runtime load libraries**

**Note:** This step is applicable only if you configured your runtime environment by using the Configuration Tool (ICAT), or if you configured your runtime environment by using the PARMGEN Workflow user interface and did not enable the optional RTE_X_STC_INAPF_INCLUDE_FLAG parameter to generate inline APF-authorize statements within the product started tasks.

Add the following runtime load libraries to your list of APF-authorized libraries.

- \&rhilev.&rte.RKANMOD
- \&rhilev.&rte.RKANMODU
- \&rhilev.&rte.RKANMODL
- \&rhilev.&rte.RKANMODP

Any other runtime libraries concatenated in the STEPLIB DDNAME or in the RKANMODL DDNAME of started tasks must also be APF-authorized.

If the runtime environment shares target libraries that were installed by SMP/E, you must also APF-authorize the following libraries:

- \&thilev.TKANMOD
Complete the configuration for the OMEGAMON subsystem (optional)

Tasks described in this section are required to complete the configuration of the optional OMEGAMON subsystem. This shared component is not used by all OMEGAMON monitoring agents. This subsystem is required if you have configured any of the following OMEGAMON monitoring agents in the runtime environment:

- IBM Tivoli OMEGAMON XE for CICS on z/OS (optional)
- IBM Tivoli OMEGAMON XE for IMS on z/OS
- IBM Tivoli OMEGAMON XE for DB2 Performance Expert/Performance Monitor on z/OS
- IBM Tivoli OMEGAMON XE for Storage on z/OS
- IBM Tivoli OMEGAMON XE on z/OS

Update the IEFSSNxx member of the SYS1.PARMLIB

If you are creating a new runtime environment, the appropriate IEFSSNxx member of SYS1.PARMLIB must be updated to identify the OMEGAMON subsystem to the z/OS system. Member KCNDLSSI (created in the Create runtime members step) in the &thilev.TKANMODL data set contains a sample IEFSSNxx update. In addition to identifying the OMEGAMON subsystem to the z/OS system, this sample causes an automatic start of the subsystem address space.

Copy the OMEGAMON subsystem started task

If you have not already done so, copy the OMEGAMON subsystem started task (CANSCN) from the RKANSAMU library to PROCLIB. You can change this name to any JCL procedure name that meets the installation standards of your site. However, do not use the name of the OMEGAMON subsystem (subsystem ID) as the name of your JCL procedure.

Update the LINKLIST

Load module KCNDLINT must be placed in an APF-authorized, link-listed library so that it is available during system IPL. Copy the module to an appropriate library in the linklist. Follow the installation standards at your site in making this decision.

Note: All runtime libraries concatenated in the STEPLIB DDNAME of the CANSCN started task must be APF-authorized.

Use the RESTART parameter

The optional RESTART parameter forces the subsystem to complete initialization by passing checks designed to prevent the start of a second address space. The RESTART parameter requires the FORCE operand, as in this example:

```
START CANSCN,RST='*,RESTART=FORCE'
```

You can stop the subsystem by issuing the z/OS STOP command, as in this example:

```
STOP CANSCN
```
Important
Use the RESTART parameter only if the subsystem address space ends abnormally and subsequent attempts to start the subsystem result in the CNDL018I message; this message indicates that the subsystem is already active. Verify that the subsystem address space named in the CNDL018I message is not active before using the RESTART parameter. If the RESTART parameter is used when the subsystem is already active, results are unpredictable. You must restart the subsystem to pick up any maintenance that was installed.

The OMEGAMON subsystem cannot be executed as a batch job; it must be invoked as a started task.

Verify the OMEGAMON subsystem installation
To verify a first-time installation, either issue the z/OS SETSSI command, or perform an IPL. If an IPL is performed, the SYSLOG contains the CNDL184I message, informing you that the OMEGAMON subsystem initialization routine has completed. If the message is not displayed after an IPL, check your update to the IEFSSNxx member of SYS1.PARMLIB.

If you chose automatic startup of the OMEGAMON subsystem address space, the following messages are displayed: CNDL001, CNDL190I, CNDL034I, and CNDL027I. If the keyword parameter form of the IEFSSNxx PARMLIB member was used, you can choose to issue the SETSSI command to define the subsystem (rather than IPL). If you choose to use the SETSSI command, or if you did not choose automatic startup at IPL, you can issue the MVS START command to start the subsystem. The following messages are displayed: CNDL001I, CNDL190I, CNDL034I, and CNDL027I.

Complete the configuration for OMEGAMON monitoring agents
Before verifying the configuration, consult the corresponding chapter within each product-specific Planning and Configuration guide to ensure that you have completed all the required steps.

Enable the self-describing agent feature or manually install application support
The Tivoli Enterprise Portal Server now supports self-describing agents, which include all the required application support files that the monitoring server needs to support the product. After startup, self-describing agents automatically push application support updates to the monitoring server, the Tivoli Enterprise Portal Server, and the Tivoli Enterprise Portal browser client. Self-describing agents eliminate the requirement to recycle the monitoring server after application support updates, and ensure that application support files are current. Note that the Tivoli Enterprise Portal desktop client is not updated through the self-describing agent process and must be updated by using the application support DVD.

If you have configured a hub monitoring server on z/OS systems, or if you have enabled the self-describing agent feature and plan to use the IBM Tivoli OMEGAMON enhanced 3270 user interface, you are not required to install a Tivoli Enterprise Portal Server for the sole purpose of adding application support files. To administer the self-describing agent feature, you must install the Tivoli Enterprise Services user interface extensions component on a distributed system. This component provides the tacmd command-line utility to interact with a monitoring server. This component is available on the IBM Tivoli Monitoring DVD image. For more information, see the IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS and the IBM Tivoli Monitoring: Command Reference.

Alternatively, you can choose to manually install application support by using the OMEGAMON Application Support DVD. You must follow these steps to install application support for any agents that do not
currently support the self-describing agent feature. Under this process, the application support is installed onto a distributed platform and then seeded to the hub monitoring server. To complete the seeding process, the hub monitoring server must be recycled. The application support for the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal clients can also be added by using the OMEGAMON Application Support DVD. Whenever you add application support using this method, the components must be recycled afterwards to complete the changes.

For detailed instructions on installing application support to a monitoring server on z/OS systems, see IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS. If your hub monitoring server is on a distributed platform, see the IBM Tivoli Monitoring: Installation and Setup Guide.

---

Install language support

If you want the Tivoli Enterprise Portal user interface, online help, and expert advice for your products to be displayed in a language other than English, you must also add language support to the portal server. For instructions on adding language support, see the IBM Tivoli Monitoring: Installation and Setup Guide.

---

Verify the configuration

After completing the post configuration tasks described within this chapter, you are ready to confirm that the products and components can run and communicate with each other.

1. If you have not already done so, run the configuration verification job in each runtime environment to confirm that all the required data sets, members, and configuration jobs were created, and that the jobs completed successfully.
   • With the PARMGEN configuration method, the configuration verification job is included in Step 8. Submit batch jobs to complete the PARMGEN setup on page 73.
   • With the Configuration Tool method, you can run the configuration verification job from the RTE Utility menu. See Runtime environment utilities on page 174.

   If the configuration verification report shows that a required item is missing or that a job failed, repeat the necessary configuration steps.

2. After you have verified all runtime environments, validate the operation of all components and the communication between them as follows:
Table 8. Validation of the runtime environment

<table>
<thead>
<tr>
<th>Configuration method</th>
<th>Steps</th>
</tr>
</thead>
</table>
| If you used the PARMGEN configuration method, a job containing all the tasks configured within the runtime environment is available to use. Complete the adjacent steps. | 1. If your hub is on a distributed system or another runtime environment, start it. For more information, see the *IBM Tivoli Monitoring: Installation and Setup Guide*.  
2. Run the composite job to start all tasks within this runtime environment:  
   \[ /s \text{xxxxSTRT} \]  
   where \text{xxxx} = \%RTE\_STC\_PREFIX\% (The default is CANS).  
   **Note:** An equivalent \text{xxxxSTOP} job is generated for stopping all tasks within the runtime environment.  
3. In the RKLVLOG for each monitoring agent address space, look for the messages that indicate successful startup. Also look for messages indicating that the monitoring agents or remote monitoring server have successfully connected to the hub, if applicable. Consult the product-specific Planning and Configuration guides for more information.  
4. If you have enabled the self-describing agent feature, check the status of the self-describing agent application support at the hub monitoring server log, monitoring agent logs and finally any remote agent logs to ensure that the self-describing agent process has completed. For more information, see the *IBM Tivoli Monitoring: Installation and Setup Guide*. |
Table 8. Validation of the runtime environment  (continued)

<table>
<thead>
<tr>
<th>Configuration method</th>
<th>Steps</th>
</tr>
</thead>
</table>
| If you prefer to start each job in turn or used the Configuration Tool (ICAT) configuration method, complete the adjacent steps. | 1. If IBM Tivoli OMEGAMON XE for Storage on z/OS is one of your installed products, start the OMEGAMON subsystem first:  
/S CANSCN  
2. Start the hub monitoring server. If your hub is on a z/OS system, start the monitoring server started task:  
/S CANSDSST  
   If your hub is on a distributed system, see the IBM Tivoli Monitoring: Installation and Setup Guide.  
3. If you have configured a remote monitoring server on a z/OS system, start its started task:  
/S CANSDSST  
   For agents that run inside the monitoring server address space, starting the monitoring server automatically starts the agents.  
4. In the RKLVLOG for each monitoring server address space, look for the messages that indicate successful startup. Also look for messages indicating that the remote monitoring server has successfully connected to the hub, if applicable.  
5. If OMEGAMON XE for Storage on z/OS is not one of your installed products and if any of your monitoring agents on z/OS require the OMEGAMON subsystem, start it now. See the configuration documentation for each monitoring agent to verify if the OMEGAMON subsystem is required.  
6. Start each monitoring agent on z/OS that runs within its own address space. Some monitoring agents may have specific instructions or a sequence in which tasks should be started. Consult the product-specific Planning and Configuration guides for information. If you have enabled the self-describing agent feature, check the status of the self-describing application support at the hub monitoring server log, monitoring agent logs and finally any remote agent logs to ensure that self-describing process has completed. For more information, see the IBM Tivoli Monitoring: Installation and Setup Guide.  
7. In the RKLVLOG member for each monitoring agent address space, look for the messages that indicate successful startup. Consult the product-specific Planning and Configuration guides for information. |
3. Ensure that data collected by the monitoring agents is available on the user interface that you have chosen to use as follows:

Table 9. Data validation on user interfaces

<table>
<thead>
<tr>
<th>User interface</th>
<th>Steps</th>
</tr>
</thead>
</table>
| If you are using the IBM Tivoli OMEGAMON enhanced 3270 user interface, complete the adjacent steps: | 1. If you have not already done so, start the OMEGAMON enhanced 3270 user interface address space: /S CANSTOM  
2. Check the SYSPRINT of the OMEGAMON enhanced 3270 user interface and look for the messages that indicate a successful startup. The following message accompanied by no error messages indicates the completion of the OMEGAMON enhanced 3270 user interface startup process:  
KOBCM0058I: Registry Refresh statistics:  
3. Log on to the OMEGAMON enhanced 3270 user interface and check that the initial workspace panels contain data sourced from the monitoring agents within this runtime environment. |
| If you are using the Tivoli Enterprise Portal, complete the adjacent steps: | 1. On a distributed system, start the Tivoli Enterprise Portal Server and client.  
2. Log on to the Tivoli Enterprise Portal client. When the Tivoli Enterprise Portal starts, check for the managed system name that you have configured (listed under the z/OS Systems entry in the Physical Navigator tree). |

**Note:** If you encounter problems, see the *IBM Tivoli Monitoring: Troubleshooting Guide* and the troubleshooting guide for each of your monitoring agents. Also see **Appendix G, “Support information,”** on page 211.

**Enable security**

After you have established that your new runtime environment is configured correctly, you can safely enable the required level of security on each of the components within your runtime environment.

If you are using the IBM Tivoli OMEGAMON enhanced 3270 user interface to view data, you can enable security by following the steps described in *Enable security on the IBM Tivoli OMEGAMON enhanced 3270 user interface* on page 112.

If you are using the Tivoli Enterprise Portal to view data, you can create user accounts that authorize users to view the monitored data and enable authentication of those accounts by enabling security through the hub monitoring server or through the portal server. For instructions on enabling authentication on a hub monitoring server on Windows, Linux and UNIX operating systems, see the *IBM Tivoli Monitoring: Administrator's Guide*. For instructions on enabling authentication on a hub monitoring server on z/OS systems, see the *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS*.

If you intend to use the OMEGAMON (3270) or OMEGAMON II user interfaces for a monitoring agent, enable security for it now. For instructions on implementing security, see the product-specific Planning and Configuration guide for each agent.
Enabling SNMP V3 passwords for autonomous agents

If you intend to use autonomous agents on a z/OS system, you can enable SNMP V3 passwords if the ICSF subsystem is available on the z/OS system.

1. From the distributed installation media for either IBM Tivoli Monitoring or IBM Tivoli Management Services on z/OS, install the Tivoli Enterprise Monitoring Agent Framework.
   
   When you select **Tivoli Enterprise Monitoring Agent Framework** during installation, the following components are installed:
   
   - Tivoli Enterprise Monitoring Agent Framework
   - IBM GSKit Security Interface
   - Tivoli Enterprise Services User Interface
   
   This software includes the itmpwdsnmp tool, which converts the passwords in SNMP trap configuration files from plain text into GSKit-encrypted passwords. For installation instructions, see *IBM Tivoli Monitoring: Installation and Setup Guide*.

2. Run the itmpwdsnmp tool on the distributed system to encrypt the passwords in an SNMP trap configuration file. For instructions, see "SNMP PassKey encryption: itmpwdsnmp" in the *IBM Tivoli Monitoring: Administrator's Guide*.
   
   If you must encrypt only a few passwords, you can run the itmpwdsnmp tool in interactive mode.

3. Upload the SNMP trap configuration file in text mode to the `&rhilev.&rte.RKANPARU` data set for the runtime environment in which the monitoring agent is configured.

4. Use one of the following methods to create a KAES256 member in the `&rhilev.&rte.RKANPARU` data set for the runtime environment in which the monitoring agent is configured.
   
   - Copy the KAES256 member from the `&rhilev.&rte.RKANPARU` data set for the runtime environment in which the monitoring server is configured to the `&rhilev.&rte.RKANPARU` data set for the runtime environment in which the monitoring agent is configured. For instructions on creating the KAES256 member, see *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS*.
   
   - In binary mode, copy the KAES256.ser file from the keyfiles directory of the distributed system where you ran the itmpwdsnmp tool to the KAES256 member of the `&rhilev.&rte.RKANPARU` data set for the runtime environment in which the monitoring agent is configured. The KAES256.ser file contains 48 bytes on distributed systems and is padded with blanks in the KAES256 member of the `&rhilev.&rte.RKANPARU` data set.

5. Concatenate the ICSF modules to the existing startup PROC `RKANMODL DDNAME` of the monitoring agent. To do so, edit the monitoring agent startup PROC and add ICSF support to the RKANMODL DDNAME. In the following example, the CSF.SCSFMOD0 data set contains the ICSF decryption modules:

   ```
   //RKANMODL DD DISP=SHR,DSN=&rhilev.&rte.RKANMODL
   // DD DISP=SHR,DSN=TDOMPT.&LVMLVL..MODL
   // DD DISP=SHR,DSN=TDOMPT.&CMSLVL..MODL
   // DD DISP=SHR,DSN=CSF.SCSFMOD0
   ```

6. Restart the monitoring agent and verify that the passwords are decrypted. Check RKLVLLOG for error messages indicating failure of GSKit password decryption or failure to create SNMP V3 trap destinations.

Enable security on the IBM Tivoli OMEGAMON enhanced 3270 user interface

To set up security for the OMEGAMON enhanced 3270 user interface, a security administrator must define a system authorization facility (SAF) general resource class and define profiles to control access to the interface, to control access to the data queries issued by the interface, and to control actions performed by the interface. Users or user groups must be given access to the profiles. And, finally, a PASSTKT profile must be defined for the OMEGAMON enhanced 3270 user interface.
To set up security for the OMEGAMON enhanced 3270 user interface, the following tasks must be completed by a security administrator:

1. Define an SAF general resource class
2. Define logon profiles to control access to the OMEGAMON enhanced 3270 user interface
3. Define Take Action profiles to control access to OMEGAMON enhanced 3270 user interface data actions
4. Define Query profiles to control access to OMEGAMON enhanced 3270 user interface data sources
5. Permit access to the profiles by appropriate personnel
6. Define a PASSTKT profile to allow the OMEGAMON enhanced 3270 user interface to propagate, identify, and perform authentication checking for query and action requests sent to the Tivoli Enterprise Monitoring Server

The OMEGAMON enhanced 3270 user interface provides a pseudo security class named OMEGDEMO. This class name is used to implement Demo mode. In Demo mode, no authorization checks are performed. Use this mode only if instructed by IBM Software Support.

To activate Demo mode, set the RTE_SECURITY_CLASS parameter to OMEGDEMO. (In the Configuration Tool, the security class is specified in the Global SAF class name field of the CREATE/UPDATE RUNTIME ENVIRONMENT panel).

If Demo mode is in effect at startup, the following messages are written to the DDNAME SYSPRINT log written to SYSOUT:

KOBCS0001A PLEASE REVIEW THE SECURITY SET UP INFORMATION
KOBCS0001A SAF SECURITY HAS NOT BEEN ENABLED/CONFIGURED
KOBCS0001A THIS PRODUCT IS RUNNING UNSECURED!!

**Define an SAF general resource class**

The resource profiles that control access to the OMEGAMON enhanced 3270 user interface are defined within an SAF general resource class. If you already have a general resource class that you want to use that has the appropriate CDT characteristics, use that class. If you do not have an appropriate class, or you want to use a different class, one must be defined.

The SAF general resource class must have the following CDT characteristics (DCT entry values):

- **CASE(UPPER)**
- **FIRST(ALPHA,NATIONAL)**
- **OTHER(ALPHA,SPECIAL,NUMERIC,NATIONAL)**
- **MAXLENGTH(246)**
- **MAXLENX(246)**
- **KEYQUALIFIERS(0)**
- **PROFILESALLOWED(YES)**
- **GENERIC(ALLOWED)**
- **RACLIST(REQUIRED)**

You can define an SAF general resource class dynamically or by updating the CDT. If you are using RACF, *special* authority is required.

- To define an SAF general resource class using the ICHERCDE macro (for RACF), code:

  ICHERCDE CLASS=classname,CASE=UPPER,DFTRETC=4,DFTUACC=NONE, FIRST=ANY,GENLIST=ALLOWED,GENERIC=ALLOWED,MAXLENX=247, OTHER=ANY,POSIT=nnn,PROFDEF=YES,RACLIST=ALLOWED
Both the classname and POSIT number should be unique. POSIT values 0 - 18, 57 - 127, and 528 – 1023 are reserved for IBM use and should not be used for your dynamic class entries. To list the POSIT numbers currently in use, you can use the following command:

```
RLIST CDT * NORACF CDTINFO
```

- To define a RACF class dynamically, use the following commands:

```
SETROPTS CLASSACT(CDT) RACLIST(CDT)
RDEFINE CDT classname UACC(NONE) CDTINFO( +
    CASE(UPPER) FIRST(ALPHA) OTHER(ALPHA,SPECIAL,NUMERIC) +
    MAXLENGTH(246) MAXLENX(246) KEYQUALIFIERS(0) +
    PROFILESALLOWED(YES) POSIT(nnn) GENERIC(ALLOWED) +
    RACLST(REQUIRED) )
SETROPTS RACLIST(CDT) REFRESH
SETROPTS CLASSACT(classname) RACLIST(classname) GENERIC(classname) REFRESH
```

**Define LOGON profiles to control access to the interface**

The authority to log on to the OMEGAMON enhanced 3270 user interface is verified by checking for access to an SAF resource named in this pattern:

```
KOB.LOGON.<plexname>.<smfid>.<stcname>
```

where KOB.LOGON is the logon resource prefix. This prefix can be changed by setting the KOB_SAF_LOGON_RESOURCE_PREFIX parameter to another value.

For example:

```
KOB_SAF_LOGON_RESOURCE_PREFIX="E3270I."
```

changes the resource name pattern for resource profiles used to control logon to the OMEGAMON enhanced 3270 user interface to

```
E3270I.<plexname>.<smfid>.<stcname>
```

- **plexname**
  - Is the name of a z/OS Sysplex.
- **smfid**
  - Is the SMF ID that represents the Sysplex member system.
- **stcname**
  - Is the job name of the OMEGAMON enhanced 3270 user interface started task.

If no matching SAF profile exists to protect an OMEGAMON enhanced 3270 user interface instance, logging on to that instance is allowed. For example, suppose you wanted to define a profile to control access to an OMEGAMON enhanced 3270 user interface started task named KOB3270 running on Sysplex IBMTEST on Sysplex member TST, in an SAF class named $KOBSEC. In this case, define a profile named KOB.LOGON.IBMTEST.TSTA.KOBE3270 by using the following commands:

```
RDEFINE $KOBSEC KOB.LOGON.IBMTEST.TSTA.KOBE3270 UACC(NONE)
SETROPTS RACLIST($KOBSEC) REFRESH
```

More generally, you can define a profile to control all logons to OMEGAMON enhanced user interfaces by using the following commands:

```
RDEFINE $KOBSEC KOB.LOGON.** UACC(NONE)
SETROPTS RACLIST($KOBSEC) REFRESH
```

**Define Take Action profiles to control access to data actions**

The authority to transmit Take Action requests from the OMEGAMON enhanced 3270 user interface to a product agent instance is verified by checking for access to an SAF resource named in this pattern:

```
<Kpp>.<msn>.TAKEACTION
```

where Kpp is the product code of the agent instance. For example, for OMEGAMON XE for CICS, the product code is KCP. For OMEGAMON XE on z/OS, the product code is KM5.
Is a managed system name. A managed system name typically identifies a unique Tivoli Enterprise Monitoring Server agent instance. Note that the form of managed system names differs from product to product. Check the agent-specific Configuration and Planning Guide for information about the form used for managed system names.

**TAKEACTION**

Is a literal. Note that unless a matching SAF profile exists to protect a given Take Action command, any request to transmit an action to the managed system name is denied. For example, suppose that you wanted to control the ability to issue a Take Action command to an OMEGAMON XE on z/OS agent running on Sysplex IBMTEST on Sysplex member TSTA, in an SAF class name $KOBSEC. In this case, you would define a profile named KM5.IBMTEST:TSTA:MVSSYS.TAKEACTION by entering these commands:

```
RDEFINE $KOBSEC KM5.IBMTEST:TSTA:MVSSYS.TAKEACTION UACC(NONE)
SETROPTS RACLIST($KOBSEC) REFRESH
```

More generally, you could define a profile to control all Take Action commands for OMEGAMON XE on z/OS:

```
RDEFINE $KOBSEC KM5.**.TAKEACTION UACC(NONE)
SETROPTS RACLIST($KOBSEC) REFRESH
```

**Define Query profiles to control access to data sources**

The authority to issue query requests from the OMEGAMON enhanced 3270 user interface to a product agent instance is verified by checking for access to an SAF resource named in this pattern:

```
Kpp.msn.tablename
```

where Kpp is the product code of the agent instance. For example, for OMEGAMON XE for CICS, the product code is KCP. For OMEGAMON XE on z/OS, the product code is KM5.

Is a managed system name. A managed system name typically identifies a unique Tivoli Enterprise Monitoring Server agent instance. Note that the form of managed system names differs from product to product. Check the agent-specific Configuration and Planning Guide for information about the form used for managed system names.

**tablename**

Is the name of the data source (attribute group, or table) defined within the product agent. Note that if a matching SAF profile does not exist to protect a given query, that query is allowed. For example, suppose you wanted to control the ability to issue a query to an OMEGAMON XE on z/OS agent running on Sysplex IBMTEST on Sysplex member TSTA, for table KM5xxxxxx. Assuming that the SAF class name is $KOBSEC, you would define a profile named KM5.IBMTEST:TSTA:MVSSYS.KM5xxxxxx by entering these commands:

```
RDEFINE $KOBSEC KM5.IBMTEST:TSTA:MVSSYS.KM5xxxxxx UACC(NONE)
SETROPTS RACLIST($KOBSEC) REFRESH
```

More generally, you could define a profile to control all data queries for a specific product:

```
RDEFINE $KOBSEC KM5.** UACC(NONE)
SETROPTS RACLIST($KOBSEC) REFRESH
```

**Permit access to profiles**

To access the profiles defined to control access, a user or group must be permitted to access the profile instances. Your security administrator will know best how to grant permissions to access these resources. As an example, assuming an SAF class name of $KOBSEC, you can allow access to a resource by entering this command:

```
PERMIT KOB.LOGON.** ID(userid) ACCESS(READ) CLASS($KOBSEC)
PERMIT Kpp.**.TAKEACTION ID(userid) ACCESS(UPDATE) CLASS($KOBSEC)
PERMIT Kpp.** ID(userid) ACCESS(READ) CLASS($KOBSEC)
```
Queries require permission for READ access. Take Actions require permission for UPDATE access. Note that if no matching SAF profile exists to protect a Take Action, that Take Action command is denied.

**Define a PASSTKT profile**

The OMEGAMON enhanced user interface uses SAF pass tickets to cross-authenticate between its interface and the hub Tivoli Enterprise Monitoring Servers. This cross-authentication allows your *session identity* to be safely passed and allows the agent instance to validate a user’s authority to request product-specific Take Action requests.

To enable cross-authentication, you must define an SAF pass ticket in each SAF security database. To enable cross-authentication for RACF, the following commands should be executed once for each RACF database instance. Do not use the KEYMASKED value shown; use one of your own. Also note that if you have more than one RACF database, the same KEYMASKED value for the KOB pass ticket definition value should be used on all RACF databases within your enterprise:

```
SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
RDEFINE PTKTDATA KOBG SSIGNON(KEYMASKED(0123456789ABCDEF))
PERMIT IRRPTAUTH.KOBG.* ID(stcuser) ACCESS(UPDATE) CLASS(PTKTDATA)
SETROPTS RACLIST(PTKTDATA) REFRESH
```

where *stcuser* is the user ID of the started task running the OMEGAMON enhanced 3270 user interface.

For the OMEGAMON enhanced 3270 user interface to propagate identity between multiple sysplexes or z/OS systems that have separate RACF databases, set the z/OS system clocks to GMT with the appropriate offset (for local time), and synchronize those system clock GMT settings as closely as possible. You must synchronize the settings because pass tickets have a time limited life span. If the GMT clock values are not synchronized, you may see SAF authentication failures (in the associated Tivoli Enterprise Monitoring Server or agent RKLVLOG).

**Enable historical data collection (optional)**

If you intend to enable historical data collection and have allocated the persistent data store, you must perform these additional tasks to complete the configuration for collecting and storing data:

1. Provide access to the persistent data store files.
2. Authorize the KPDDSCO module.
3. Verify persistent data store configuration on page 117.
4. Enable historical data collection within the Tivoli Enterprise Portal on page 117

If you are upgrading an existing monitoring server or monitoring agent, you must also refresh the KPDPROC1 maintenance procedure in your system procedure library. For more information, see *IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Upgrade Guide*.

**Provide access to the persistent data store files**

Ensure that the KPDPROC1 procedure has the required authority to read, write, and update the persistent data store files. The KPDPROC1 procedure is used to maintain the physical files that constitute the persistent data store. Data store files are archived, exported, or recycled according to the maintenance strategy that you specify for persistent data store file groups for the product. The persistent data store subsystem automatically submits maintenance jobs whenever a data store file becomes full. The maintenance procedure must be available in a system procedure library for the procedure to operate. The procedure is generic and can be used by all runtime environments using this version of the persistent data store.

**Authorize the KPDDSCO module**

The KPDDPROC REXX procedure runs in a TSO environment and must be enabled to run as an authorized program under TSO. Authorize the KPDDSCO module by adding KPDDSCO to the system
SYS1.PARMLIB(IKJTSO\nn) under the AUTHPGM section and refresh the IKJTSO\nn member by issuing the set command (T IKJTSO=\nn). You might also request that authorized system programmers perform this step so it can be scheduled with the LPAR change control processes.

Verify persistent data store configuration
To verify that the configuration and authorization of the procedures have been successful, perform the following steps:

1. Bring up the started task (for monitoring server or monitoring agent) that will collect historical data into the product's persistent data store libraries. In the RKPDLLOG DDNAME started task, find any persistent data store libraries in a non-Offline status (for example, Partial or Full status).

2. From a z/OS operator console, issue the following z/OS MODIFY command:
   `/F &stcname,KPDCMD RECOVER FILE=DSN:&pds_data set`
   where &stcname is the name of the started task performing the persistent data store collection, and &pds_data set is the persistent data store data set.
   For example, issue the following MODIFY command for the monitoring server:
   `/F CIDSST,KPDCMD RECOVER FILE=DSN:rhilev.&rte.RGENHIS1`
   DSN:rhilev.&rte.RGENHIS1

3. Wait 5 minutes.

4. In the RKPDLLOG DDNAME started task, find the following Command: and KPDDSTR: references as shown in the following monitoring server RKPDLLOG DDNAME example:
   Command: RESUME FILE=DSN:rhilev.&rte.RGENHIS1
   KPDDSTR: CONNECT processing started for DataStore file
   DSN:rhilev.&rte.RGENHIS1
   KPDDSTR: CONNECT processing ended for DataStore file
   DSN:rhilev.&rte.RGENHIS1

5. If these references are not found, view the KPDPROC1 started task in SDSF and look for any obvious errors.

Enable historical data collection within the Tivoli Enterprise Portal
No historical data is collected unless the collection of historical data is enabled in the Tivoli Enterprise Portal.

The Tivoli Enterprise Portal History Collection dialog box lists the attribute groups, or tables, for each monitoring agent that are enabled for historical collection and reporting. To enable historical data collection, you must select and configure each attribute table for which you want to collect data, and then start collection for those tables. If you want to warehouse the data for long-term historical reporting, you must set the Warehouse Interval to the interval at which data is warehoused.

For detailed instructions on configuring history data collection, see the IBM Tivoli Monitoring: Administrator's Guide and the Tivoli Enterprise Portal online help.

Run the ITMSUPER Tools (optional)
Use the ITMSUPER Tools to learn about the health of your managed systems, situations, and environment configuration. The ITMSUPER Tools are included in the IBM Support Assistant (ISA), a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. To install the ISA software, go to [http://www.ibm.com/software/support/isa](http://www.ibm.com/software/support/isa).
Part 3. Configuration Tool (ICAT) scenarios

This part of the Common Planning and Configuration Guide contains information to help you replicate configured runtime environments and complete an enterprise-wide deployment.

- For the PARMGEN configuration method, follow the instructions in Chapter 4, “Configuring products using the PARMGEN Workflow user interface,” on page 55.
- For the Configuration Tool (ICAT) method, follow the instructions in Chapter 7, “Using the Configuration Tool (ICAT) batch mode to replicate a configured environment,” on page 121 and Chapter 8, “Configuration Tool (ICAT) deployment scenarios,” on page 139.
- For both methods, see Appendix A, “Enabling system variable support,” on page 165 as needed.
Chapter 7. Using the Configuration Tool (ICAT) batch mode to replicate a configured environment

After you have used the Configuration Tool in interactive mode to configure a runtime environment, you can use the tool in batch mode to replicate that environment to other z/OS images. Using batch mode, you can install a complete runtime environment by submitting a single batch job, rather than configuring it panel by panel.

This section focuses on the use of batch mode to create new runtime environments based on a model environment. However, the Configuration Tool batch mode facilities can be used for several different purposes:

- Backing up the configuration parameters of an existing runtime environment.
- Restoring a runtime environment based on a saved configuration.
- Reconfiguring an existing runtime environment when something in the environment has changed (such as a port or subsystem name).
- Creating a new runtime environment based on an existing one.

The principal components of batch mode processing are the batch mode job and the batch parameter member. The batch mode job validates the input configuration parameters and generates the jobs required to build, configure, and load a runtime environment. The job uses a single parameter member containing all the configuration parameters as input for the jobs it generates. The parameter member is derived from an existing runtime environment using a Configuration Tool utility. You can adjust the parameters before using the parameter member to create a new runtime environment or to reconfigure or restore an existing one.

The topics in this section describe the tasks to be performed for batch mode processing in order:

- **“Step 1. Create the batch mode job” on page 122**
  The batch mode job, CICATB, is used to create and submit the jobs that build, configure and load a runtime environment using parameters exported from an existing, or model, runtime environment. It can also be used to validate the parameters. The batch mode job does not exist by default; you must generate it by using the Configuration Tool's Create Batch Mode Job service. Creating the batch mode job also creates the KCISETUP job, which you invoke to set up the environment required by the batch utilities.

- **“Step 2. Create and clone a batch mode parameter member” on page 125**
  A batch mode parameter member contains the configuration values for all the products configured in an existing runtime environment. The member is created by using the Create/Clone Batch Mode Parameters utility to export the configuration values into a library member. You clone the member and modify the values as required to create a new runtime environment.

- **“Step 3. Adjust and validate the parameters” on page 128**
  If you do not want to create an exact copy of an existing runtime environment, you adjust the configuration parameters in the cloned parameter member. You validate those changes by using the SCAN function of the CICATB job.

- **“Step 4. Generate sample transport JCL” on page 130**
  The Configuration Tool provides several sample DFDSS tape jobs that you can use to transport the runtime environment or the configuration jobs or parameter member required for creating the runtime environment to the target system. It also provides sample scenarios for using the jobs.

- **“Step 5. Create a new runtime environment” on page 131**
  You can create new runtime environments on target z/OS images using one of several methods.

- **“Step 6. Complete the configuration of a new runtime environment” on page 137**

© Copyright IBM Corp. 2008, 2012
You must complete steps outside the Configuration Tool on the target image just as you did on the model image. This section explains how to find information about the required steps.

**Tips**
- Read this entire chapter before you begin performing the tasks described.
- If you intend to use batch mode processing to replicate your runtime environments, enable system variable support when you configure the model runtime environment. System variable support enables runtime environments to inherit the system values for the systems where they are defined. See Appendix A, “Enabling system variable support,” on page 165 for detailed instructions.

### Step 1. Create the batch mode job

The batch mode job, CICATB, generates the jobs that create the replicated runtime environment. CICATB can also be used to validate the parameters in a parameter member.

There is no default batch mode job. You create one by running the Create Batch Mode Job service. Running Create Batch Mode Job also generates KCISETUP, which is invoked to set up the Configuration Tool environment required by the batch utilities. (See KCISETUP: Set up the environment for batch utilities on page 178 for more information.)

**Tip:** After KCISETUP and CICATB are created, they can be used for all subsequent batch parameter member processing unless the ISPF environment changes. If the ISPF environment changes, recreate KCISETUP and CICATB before you do any further batch processing.

You must exit the Configuration Tool before the CICATB job can run.

- “Creating the batch mode job”
- “Batch mode job submission options” on page 123
- “Batch mode configuration jobs” on page 124
- “Batch mode job report” on page 124

#### Creating the batch mode job

Follow this procedure to create the batch mode job.

1. Start the Configuration Tool by entering the following command in a TSO command panel:
   ```
   ===> EXEC 'shilev.INSTLIB'
   ```
2. From the Main Menu, select Option 1 (Set up work environment) and press Enter.
   The Set up work environment panel is displayed.
3. Select option 3 (Create batch mode job).
   The Create batch mode job option generates the CICATB JCL, which runs the configuration process in batch mode. This option also creates the KCISETUP REXX exec. When invoked, this exec enables ISPF to use macros supplied by the Configuration Tool.
   The CICATB job is created in the &shilev.INSTJOBS data set. You use this job to create new runtime environments with the configuration values from the original production runtime environments.
Tips

- To see the complete path of the library the batch mode job was created in, press F1. A text box at the bottom of the panel shows the complete path. The text box is displayed only if F1 is pressed immediately after Create batch mode job is selected. Otherwise, the general help for the panel is displayed.

- When you create the CICATB batch job member in the &shilev INSTJOBS data set, the Configuration Tool also creates the batch jobcard ISPF table KCITPIG1 in the &shilev INSTDATA data set. The contents of the KCITPIG1 table are used as the standard jobcard in batch configuration jobs. If you modify the jobcard on the Specify JCL Options panel (Figure 29 on page 80), or if the ISPF environment changes in some other manner, recreate the CICATB batch job to refresh the KCITPIG1 jobcard table before running CICATB. When the ISPF environment changes, CICATB must be recreated to modify the DDNAME concatenation accordingly, so that the job no longer points to the old ISPF libraries.

4. Wait until you see the CICATB JOB CREATED message in the upper-right corner of the panel.
5. Open the CICATB JCL and examine it. The SYSTSIN DD statement in the JCL contains several parameters that you must customize before you submit the job.

**BATCHLIB**
The library that contains the batch runtime environment parameter member. The default is &shilev INSTJOBS.

**BATCHMEM**
The member name of the batch runtime environment parameter member (the name must be the same as the name of the runtime environment you intend to create). You create this member in the following step.

**SUBMIT**
The type of processing. You specify different dispositions depending on the step you are performing and the requirements of the environment. See "Batch mode job submission options."

**Important**
Do not submit the CICATB job yet. Read the rest of the chapter first. Your choice of a transport method ("Step 5. Create a new runtime environment" on page 131) and related decisions affect the other tasks to be performed.

Batch mode job submission options

The SUBMIT parameter in the CICATB job specifies a disposition for runtime environment configuration jobs generated by CICATB. The valid values are:

**YES**
Create and submit the runtime environment configuration jobs. If the jobs already exist, such as when you are reconfiguring the runtime environment, the jobs are regenerated but are not submitted for execution. If you want to submit the jobs in such an environment, use the RERUN option.

If your site uses a batch job utility that analyzes the submitted JCL before the job is sent to the execution queue, use the YES option only if the jobs are single-threaded through the analyzer. Otherwise, SUBMIT(YES) might cause CICATB-generated jobs to be executed out of sequence.
**Important**

If the runtime environment contains any user-modified elements that will be modified by any configuration job generated by the CICATB job, running CICATB creates reports identifying those elements. Each report is named for the job that will modify user-modified elements: If a job named `pp%nssss` will modify user-modified elements, a report is stored in `pp%nssss`. Examine the reports and the user-modified elements they identify before running the configuration jobs. The first time you run the CICATB job, it is best to specify `SUBMIT(NO)` and examine the report on user-modified elements before submitting the configuration jobs. If the runtime environment contains any user-modified elements and if you intend to run the CICATB job with `SUBMIT(RERUN)` or `SUBMIT(YES)` specified, back up the runtime libraries before running CICATB.

RERUN  Create and submit the runtime environment configuration jobs, regardless of whether the jobs already exist. Do not use this option if the runtime environment contains any user-modified elements. Otherwise, use it to avoid deleting the configuration jobs in the `&shilev.INSTJOBS` data set if the job names already exist, before reconfiguring the runtime environment in batch mode.

NO  Create but do not submit the runtime environment configuration jobs. Used when the runtime environment will be created on a different z/OS image, or when manual control over job submission is desired.

SCAN  Validate the input parameter member and produce a report. The scan validates that the required variables are defined, and that the defined values meet the syntax requirements for the respective variables.

SCAND  Scan the input parameter member and produce a detailed data dictionary of all the runtime environment parameters. The dictionary provides a description and syntax rules for each parameter.

For more information on the SCAN and SCAND dispositions, see “Step 3. Adjust and validate the parameters” on page 128. See the CICATB job itself for information on other input control parameters.

**Batch mode configuration jobs**

If there are no parameter errors, running CICATB with either `SUBMIT(YES)`, `SUBMIT(RERUN)`, or `SUBMIT(NO)` produces a set of runtime environment configuration jobs named `pp%nssss`.

where:

- `pp` is the 2-character product code (DS is the product code for the Tivoli Enterprise Monitoring Server).
- `n` is the number of the job
- `ssss` is the JCL suffix for the runtime environment

The configuration jobs are saved in the `&shilev.INSTJOBS` data set.

**Batch mode job report**

Part of the CICATB job output is a report that has several sections, depending on the disposition and outcome. The report is in the KCIPMRPT DDNAME of the CICATB job output. In addition to KCIPMRPT, the CICATB report is also saved as a member in the `&shilev.INSTJOBS` library. The name of the report member depends on the disposition of the `SUBMIT` command in CICATB job, as shown in Table 10 on page 125.
Table 10. CICATB job reports

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Result</th>
<th>Report name</th>
<th>Report contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBMIT(SCAN)</td>
<td>No errors</td>
<td>CB#Sssss</td>
<td>Two sections:</td>
</tr>
<tr>
<td></td>
<td>Errors</td>
<td>CB#S</td>
<td>1. Parameter errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Parameters you have changed</td>
</tr>
<tr>
<td>SUBMIT(SCAND)</td>
<td>Any</td>
<td>CB#SDICT</td>
<td>Detailed parameter dictionary with four sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. List of parameters and descriptions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. List of parameters, variable names, and default values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. List of parameters, lengths, types, and subtypes, with indication of whether CAPS(ON) or CAPS(OFF) is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. List of numeric parameters, with minimum, maximum, and default values.</td>
</tr>
<tr>
<td>SUBMIT(NO),</td>
<td>No errors</td>
<td>CB#Rssss</td>
<td>Five sections:</td>
</tr>
<tr>
<td>SUBMIT(YES),</td>
<td></td>
<td>CB#R</td>
<td>1. Parameter errors.</td>
</tr>
<tr>
<td>SUBMIT(RERUN)</td>
<td>Errors</td>
<td>CB#R</td>
<td>2. Parameters you have changed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Jobs sorted by generation sequence.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. Jobs sorted by member name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5. Components configured in the runtime environment.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When there are errors, sections 3, 4, and 5 are empty.</td>
</tr>
</tbody>
</table>

Step 2. Create and clone a batch mode parameter member

A batch mode parameter member contains configuration values extracted from an existing runtime environment. The Configuration Tool creates, or regenerates, a parameter member each time the runtime environment Load function runs. This member can be used as a backup from which the current runtime environment can be recreated. The batch mode parameter member can also be cloned, modified if necessary, and used with the batch mode job to create a new runtime environment.

You clone a batch mode parameter member by using the Create/Clone batch mode parameters utility. The name you give to the new member becomes the mid-level qualifier for the new runtime environment you create by using the member.

The Create/Clone batch mode parameters utility offers three options:

1. **Create batch mode parameter deck.**
   
   Use this option to create or recreate a batch mode parameter deck for the selected runtime environment. You might use this option, for example, if you want to create a parameter deck without loading the runtime libraries.

2. **Clone batch mode parameter deck, unmodified.**
   
   Use this option to create a clone of a batch mode parameter deck that includes all the products and components in the runtime environment. If a parameter deck does not already exist, the utility creates one and then creates the clone.

3. **Clone batch mode parameter deck, with modifications.**
Use this option if you want to create a clone of a batch mode parameter deck, but you do not want to include all of the products configured in the runtime environment in the deck. During the cloning process, you are presented with a panel displaying a table of all of the components configured in the RTE, from which you can select components to be excluded from the cloning process.

Every parameter deck requires a unique 1- to 4-character runtime environment JCL suffix. During the cloning process, you are prompted to enter one for the clone member. The clone member will contain this new value.

Complete the following steps to generate and clone a parameter member:

1. Start the Configuration Tool by entering the following command in a TSO command panel:
   ```
   ===> EXEC 'shilev.INSTLIB'
   ```

2. Navigate to the Runtime Environments (RTEs) panel:
   a. On the Main Menu panel select option 3 (Configure products).
      The Configure Products panel is displayed.
   b. Select option 1 (Select product to configure).
      The Product Selection Menu panel is displayed.
   c. Select a product that is configured in the runtime environment from which you want to export the parameters.
      The Runtime Environments (RTEs) panel is displayed.

   **Tip:** Enter `panelid` in the `OPTION ===>` line to see the panel ID in the upper left corner of the panel.

3. Type `Z` next to the runtime environment you want to replicate and press Enter.
   The RTE Utility Menu is displayed.

4. Select option 1 (Create/Clone batch mode parameters) and press Enter.
   The Create/Clone Parameter Deck Menu is displayed (see Figure 45 on page 152).

5. Select option 2 (Clone batch mode parameter deck, unmodified) to create a cloned parameter member that contains all the same products as the model runtime environment, or select option 3 (Clone batch mode parameter deck, modified) to create a clone parameter member that excludes some of the products in the model runtime environment.

6. Press Enter. The Create/Clone Batch Mode Parameters panel is displayed (see Figure 40).

   **Figure 40. Create/Clone Batch Mode Parameters panel**
   
   The panel contains default values for the input parameter member for the existing runtime environment. By default, the name of the parameter member is the name of the runtime environment and it is created in the `&shilev.INSTJOBS` library. You can edit these values if you want. If the parameter deck does not already exist, the utility creates it.
Tip: The cloned parameter member (Output parameter deck member) is a copy of the original (Input parameter deck member). If you provide a different value for the output Parameter library or for the Clone member name, creating the clone does not destroy the original.

7. Specify appropriate values for the Output parameter deck member section of the panel:

Parameter library
The library where the clone parameter member will be created. By default, the library is &shilev.INSTJOBS. You can change the library name, but the library that you specify must already exist. If the library does not exist, the Configuration Tool does not create the parameter member and display the message LIBRARY ERROR in the upper right corner of the panel.

Clone member name
The name of the member that contains the parameters. This name becomes the mid-level qualifier for the new runtime environment.

Edit after create
Y or N, to specify whether the clone parameter member is to be opened for editing after it is created.

8. Press Enter.

If the input parameter deck member does not already exist, the following message is displayed:
The parameter deck member rte does not exist.
Press Enter to continue and create rte parameters or F3 to cancel.

9. Press Enter to create the parameter deck.

After parameter member for the model runtime environment is created, the following message is displayed:
BATCH PARMS CREATED

10. Press Enter again to generate the clone parameter deck.

The following message is displayed:

```
------- Change RTE JCL suffix panel ------

The cloned batch parameter deck includes a value for the RTE JCL suffix.
Every RTE should have a unique one to four character RTE_JCL_SUFF value.
The RTE_JCL_SUFF value for the original RTE SYSA = SYSA
Specify a new unique RTE_JCL_SUFF for RTE SYSB ==> ____
```

11. Type the JCL suffix for the new runtime environment and press Enter.

If you specified Y for Edit after create, the clone member is displayed.

12. Edit the new library member to reflect the settings specific to the z/OS image where the new runtime environment will be created, as discussed in “Step 3. Adjust and validate the parameters” on page 128.
Tip

Use the KCICGPHP utility to display help for the parameters while editing the batch parameter member. Use either of the following methods:

- From an ISPF Edit command-line, enter the command KCICGPHP, position the cursor on the row that contains a batch parameter, and then press Enter.
- Position your cursor on the row that contains a batch parameter and then select the PF key assigned by the Configuration Tool PF Key Manager.

For more information on setting up and using the help, see "KCICGPHP: Display help for configuration parameters" on page 180

13. Press F3 to save the changes and close the parameter member. After the clone parameter member for the target runtime environment is created, the following message is displayed:

    RTE CLONED

Step 3. Adjust and validate the parameters

The parameters in the clone batch mode parameter member describe the configuration of the model runtime environment from which they were exported. You might want to adjust the parameter values for the new environment.

Compare the values that were used for the model runtime environment with the values of the new runtime environment. (You can use a scan report on the parameter member for the model environment to review the values.) If a variable value must be changed for the new runtime environment (for example, the DB2 subsystem name or z/OS hostname), open the clone parameter member in edit mode. Adjust the parameters as necessary and save the changes.

After modifying the values, validate the parameters in the parameter member before you create the new runtime environment by running the CICATB job with SUBMIT(SCAN). The job checks, for example, that a numeric parameter is in the allowed range, or that the length of a character parameter does not exceed the maximum number of allowed characters. Some validations performed by the Configuration Tool in interactive mode are not performed in batch mode.

To validate the parameters:

1. If the CICATB job is not already open in the Configuration Tool, open it from the library in edit mode.
2. Scroll down until you come to the SYSTSIN section.
3. Replace the default value for BATCHMEM with the name of your new parameter member as shown in the following example. Set the disposition for SUBMIT to SCAN (the default).

   //SYSTSIN DD *,DCB=BLKSIZE=80
   PROFILE PREFIX(TST)
   ISPSTART +
   CMD(EX 'shilev.INSTLIB(KINSTALL)' +
        'INSTLIB(shilev.INSTLIB) +
        BATCHLIB(shilev.INSTJOBS) +
        BATCHMEM(SYSB) + Replace name of current with name of new parameter member
        SUBMIT(SCAN))
   /*
   */
4. Save the changes.
5. Submit the job.
6. Press F3 to leave the CICATB job. If you are editing the job in INSTJOBS in the Configuration Tool, exit the tool. Otherwise the job can not continue.
7. Switch to SDSF and check the status of your job. If it has finished, use `?` to select it and see the separate logs. Check SYSTSPRT first for error messages.

It is not sufficient to look at the return code of the job. This will be 0 even if errors occurred. Check SYSTSPRT. If the scan did not find any error in the parameters, the scan is successful and the SYSTSPRT looks similar to the one in Figure 41:

---

```plaintext

***********************************************************************
TOP OF DATA ***************************************************************
READY
PROFILE PREFIX(TST)
READY
ISPSSTART CMD( EX 'shilev.INSTLIB(KINSTALL)' 'INSTLIB(shilev.BATCHINSTLIB) BATCHL
EM(rte) SUBMIT(SCAN)')
*** BATCH: PRODUCT LIST IS D2621
*** 14:53:05 JOB: CB#STST DESC: BATCH PARAMETER INPUT SCAN
*** RTE.JOB09598.D0000104.? was preallocated (no free was done).
READY
END
***********************************************************************

Figure 41. SYSTSPRT for successful scan

The scan report is in KCIPMRPT. Additionally, if the scan runs with no errors, a report named CB#ssss is generated in the INSTJOBS library (where `ssss` is the JCL suffix for the runtime environment). If errors are found, a report named CB#ssss is generated.

If you do not get the success message in SYSTSPRT, the report KCIPMRPT is not generated. Check for one of the following errors:

**BATCHMEM Parameter Value State Error**

The parameter member was not found. Most likely the default value was not changed in the CICATB job. Open CICATB and check that the data set in BATCHMEM exists in the library that is specified in BATCHLIB.

**INSTLIB Parameter Value State Error**

The INSTLIB library was not found. Open CICATB and correct the path name.

**KCICCSTR NO PRODUCTS**

The target libraries that were installed by SMP/E were not found. Navigate to the Set Up Configuration Environment panel (from the Main Menu select **Set up work environment**, and then select **Set up configuration environment**) and verify that the value for SMP/E data sets **Target** is set correctly.

**SCANNER ERROR cccc FROM KCICRTE**

A SCANNER ERROR indicates that a parameter did not pass validation. Open the report KCIPMRPT and look at Section 1. The parameter that causes the error is listed there. For example:

```plaintext

***********************************************************************
SECTION 1: PARAMETER ERRORS
***********************************************************************
PARAMETER INPUT ERROR
NAME LINE# DESCRIPTION
-----------------------------------------------------
KDS_CMS_TCP_PIPE_PORT 151 Invalid Parameter Value: Value
greater-than allowable maximum of 65535

Edit the batch parameter member in INSTJOBS to change the value to an allowed value. Save the changes and run the CICATB job again.
```
If you are not sure what value a variable can have (for example, the allowed value range for a number or the allowed number of characters for a string) you can use the detailed data dictionary report of CICATB.

To generate the detailed data dictionary report:
1. Open the CICATB job in edit mode.
2. Change the disposition for SUBMIT to SCAND.
3. Save the changes and submit the job.
4. If CICATB is in the INSTJOBS library, press F3 to leave the CICATB job and press F3 again to exit the browse mode of the INSTJOBS library. Otherwise the job can not continue.
5. Start SDSF and check the status of your job.
   If it has finished, select it with ? to see the separate logs.

The detailed data dictionary report is in KCIPMRPT. Additionally, the report is saved in the CB#SDICT member of the &shilev.INSTJOBS data set.

For questions on allowed characters per string go to

SECTION 3: LENGTHS, TYPE, SUBTYPE, CAPS

For questions on allowed value range for numbers go to SECTION 4:

SECTION 4: NUMERIC PARAMETERS WITH MIN, MAX, AND DEFAULTS

and search for your variable.

After you have fixed the problem, rerun the CICATB job with SUBMIT(SCAN). Before you move on to the next step, make sure that the validation does not report any more errors.

---

**Step 4. Generate sample transport JCL**

The Configuration Tool provides sample configuration scenarios and DFDSS tape jobs for transporting the runtime environment or the components required for creating the runtime environment to the target system.

Take these steps to generate the samples:
1. If necessary, start the Configuration Tool by entering the following command in a TSO command panel:
   ```
   ===>
   EXEC 'shilev.INSTLIB'
   ```
2. Navigate to the Runtime Environments (RTEs) panel:
   a. On the Main Menu panel select option 3 (Configure products).
      The Configure Products panel opens.
   b. Select option 1 (Select product to configure).
      The Product Selection Menu panel opens.
   c. Select a product that is configured in the runtime environment from which you want to export the parameters. The Runtime Environments (RTEs) panel opens.
3. Type Z next to the runtime environment you want to generate the samples for and press Enter.
   The RTE Utility Menu panel opens.
4. Select 5 (Generate sample transport JCL) and press Enter.
   This action causes several sample transport jobs to be generated in the &rhilev.&rte.RKANSAMU library.

The generated members include:
Step 5. Create a new runtime environment

Use any of the following methods to create a new runtime environment on the appropriate z/OS image:

- "Scenario A: Defining a runtime environment on a z/OS image using shared DASD" on page 133
- "Scenario B: Transporting a runtime environment from one z/OS image to another" on page 133
- "Scenario C: Transporting runtime environment batch jobs from one z/OS image to another equipped with the Configuration Tool" on page 134
- "Scenario D: Transporting runtime environment batch mode parameters from one z/OS image to another" on page 136

These methods and their advantages and disadvantages are summarized in Table 11 on page 132
<table>
<thead>
<tr>
<th>Method</th>
<th>This method uses</th>
<th>Advantages and disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining a runtime environment on a z/OS image using shared DASD</td>
<td>Interactive and batch modes</td>
<td>Advantages of this method:</td>
</tr>
<tr>
<td></td>
<td>The Configuration Tool on the master z/OS image to create a runtime environment</td>
<td>• The Configuration Tool located on the master z/OS image contains the configuration information for all images.</td>
</tr>
<tr>
<td></td>
<td>accessible to the target image.</td>
<td>• Only one copy of the runtime libraries is created.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Only one batch job is submitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The disadvantage of this method is that it is applicable only to z/OS images with shared DASD.</td>
</tr>
<tr>
<td>Transporting a runtime environment from one image to another</td>
<td>Interactive and batch modes</td>
<td>Advantages of this method:</td>
</tr>
<tr>
<td></td>
<td>The Configuration Tool on the first image to create a runtime environment.</td>
<td>• The Configuration Tool on the first image contains the configuration information for all images.</td>
</tr>
<tr>
<td></td>
<td>After the runtime environment has been defined, you use sample transport jobs to</td>
<td>• Only one batch job is submitted.</td>
</tr>
<tr>
<td></td>
<td>ship the runtime libraries and parameters to other images.</td>
<td>• This method can be used for configuring the other z/OS images that do not share DASD.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The disadvantage of using this method is that multiple copies of the runtime libraries are created.</td>
</tr>
<tr>
<td>Transporting runtime environment batch jobs from one z/OS image to</td>
<td>Interactive and batch modes</td>
<td>Advantages of this method:</td>
</tr>
<tr>
<td>another</td>
<td>The Configuration Tool on the first image to create a set of batch jobs that can</td>
<td>• The Configuration Tool on the first image contains the configuration information for all images.</td>
</tr>
<tr>
<td></td>
<td>build a runtime environment. After the jobs are created, you use sample transport</td>
<td>• This method can be used for configuring the other z/OS images that do not share DASD.</td>
</tr>
<tr>
<td></td>
<td>jobs to ship the batch jobs to other images. The jobs are manually submitted on</td>
<td>• Only one copy of the runtime libraries is created.</td>
</tr>
<tr>
<td></td>
<td>the other images to create the runtime libraries and parameters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The disadvantage of using this method is that you must manually submit a series of batch jobs, or run a product-provided CLIST to submit the jobs that will create the runtime environment.</td>
</tr>
<tr>
<td>Transporting runtime environment batch mode parameters from one z/OS</td>
<td>Interactive mode</td>
<td>Advantages of this method:</td>
</tr>
<tr>
<td>image to another equipped with the Configuration Tool</td>
<td>The Configuration Tool in interactive mode on the first image to export an existing</td>
<td>• This method can be used for configuring the other z/OS images that do not share DASD.</td>
</tr>
<tr>
<td></td>
<td>runtime environment. After the runtime environment parameters have been</td>
<td>• Only one copy of the runtime libraries is created.</td>
</tr>
<tr>
<td></td>
<td>collected, you use the sample transport jobs to ship the batch mode parameters to</td>
<td>• Only one batch job is submitted.</td>
</tr>
<tr>
<td></td>
<td>other images. Then you run the Configuration Tool in batch mode on the other</td>
<td></td>
</tr>
<tr>
<td></td>
<td>images to create the runtime libraries and parameters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The disadvantage of using this method is that the Configuration Tool located on the first image does not contain the configuration information for all images.</td>
</tr>
</tbody>
</table>
Before using the procedures in the following sections, make sure that sufficient space and library security authorizations exist on the target systems. These procedures assume that you have already set up the batch processing environment and created the batch processing job on the first image, as described in "Step 1. Create the batch mode job" and generated the batch mode parameter member, as described in "Step 2. Create and clone a batch mode parameter member." The scenarios in this section use the sample DFDSS tape jobs created by the Configuration Tool, but tape is not the only mode of transport you can use between systems that do not share DASD. If the target system is connected to the network, you can modify the samples and dump the files to data sets, from which the files (and the jobs to restore them) can be sent to the target system through the network by FTP or TSO SEND/RECEIVE.

Consult the configuration guides for the products configured in the runtime environment for post-configuration steps that might be required on the target system.

**Scenario A: Defining a runtime environment on a z/OS image using shared DASD**

Follow these steps to define a runtime environment on a z/OS image using shared DASD.

1. Exit the Configuration Tool.
2. If you have not already done so, perform a scan on your runtime environment parameters:
   a. Edit CICATB, updating the BATCHLIB and BATCHMEM parameters as necessary, and setting the SUBMIT parameter to SCAN.
   b. Submit the CICATB job to scan your runtime environment parameters.
   c. Exit the &shilev.INSTJOBS data set so that CICATB can run.
   d. Verify that the job completes successfully, review the parameter report, correct any errors in the parameter member, and repeat the scan until a clean report is generated.
3. Create a new runtime environment that is accessible to the target image:
   a. Edit CICATB again, setting the SUBMIT parameter to YES.
      Setting SUBMIT(YES) automatically submits the configuration jobs that allocate and populate runtime libraries, unless the jobs already exist in the &shilev.INSTJOBS data set. If you are using batch processing in a cloned environment, you can specify SUBMIT(RERUN) to submit the configuration jobs automatically even if the jobs already exist.
      If the runtime environment contains user-modified elements, back up the runtime libraries before submitting CICATB with SUBMIT(RERUN) or SUBMIT(YES) specified.
   b. Submit the CICATB job to create the runtime environment.
   c. Exit the &shilev.INSTJOBS data set so that CICATB can run.
   d. Verify that the job completes successfully.
4. Perform any manual configuration steps on the target image, such as:
   - Copying procedures to PROCLIB
   - Copying VTAM definitions to VTAMLST
   - APF-authorizing libraries

See "Step 6. Complete the configuration of a new runtime environment" for more information.

**Scenario B: Transporting a runtime environment from one z/OS image to another**

Follow these steps to create a runtime environment and transport it from one z/OS image to another:

1. Exit the Configuration Tool.
2. If you have not already done so, perform a scan on the parameters for the new runtime environment:
a. Edit CICATB, updating the BATCHLIB and BATCHMEM parameters as necessary, and setting the
SUBMIT parameter to SCAN.
b. Submit the CICATB job to scan your runtime environment parameters.
c. Exit the &shilev.INSTJOBS data set so that CiCATB can run.
d. Verify that the job completes successfully, review the parameter report, correct any errors in the
parameter member, and repeat the scan until a clean report is generated.

3. Create a new runtime environment that is accessible to the target image runtime environment:
a. Edit CICATB again, setting the SUBMIT parameter to YES. This submits the runtime environment
configuration jobs that allocate and populate runtime libraries.
b. Submit the CICATB job to create the runtime environment.
c. Verify that the job completes successfully.

4. Start the Configuration Tool again.

5. Navigate to the Runtime Environments (RTEs) panel:
a. On the Main Menu panel select option 3 (Configure products).
   The Configure Products panel is displayed.
b. Select option 1 (Select product to configure).
   The Product Selection Menu panel is displayed.
c. Select a product that is configured in the model runtime environment that you are going to
   replicate. The Runtime Environments (RTEs) panel is displayed.

6. Generate the sample transport JCL:
a. Type Z next to the runtime environment you want to transport and press Enter.
   The RTE Utility Menu panel is displayed.
b. Select 5 (Generate sample transport JCL) and press Enter.
   This action causes several sample transport jobs to be generated in the &rhilev.&rte.RKANSAMU
   library. For more information about these jobs, see “Step 4. Generate sample transport JCL” on
   page 130.

7. Use the sample jobs appropriate to this scenario:
   • Use sample job XDFDMP01 on the first image to dump the runtime libraries
   • Use sample job XDFRST01 on the target image to restore the runtime libraries

8. After you have transported the runtime environment to the target images, perform any required manual
configuration steps on the target image, such as:
   • Copying procedures to PROCLIB
   • Copying VTAM definitions to VTAMLST
   • APF-authorizing libraries
   See “Step 6. Complete the configuration of a new runtime environment” on page 137 for more
   information.

Scenario C: Transporting runtime environment batch jobs from one
z/OS image to another equipped with the Configuration Tool

Follow these steps to transport runtime environment batch jobs from one z/OS image to another equipped
with the Configuration Tool.

1. Exit the Configuration Tool.
2. If you have not already done so, perform a scan on your runtime environment parameters:
a. Edit CICATB, updating the BATCHLIB and BATCHMEM parameters as necessary, and setting the
   SUBMIT parameter to SCAN.
b. Submit the CICATB job to scan your runtime environment parameters.
c. Verify that the job completes successfully, review the parameter report, correct any errors in the parameter member, and repeat the scan until a clean report is generated.

3. Create the runtime environment generation jobs:
   a. Edit CICATB again, setting the SUBMIT parameter to NO.
      This creates the runtime environment configuration jobs that allocate and populate runtime libraries, but does not run the jobs.
   b. Submit the CICATB job.
   c. Verify that the job completes successfully.
      You have now successfully created a set of runtime environment configuration batch jobs that must be transported to the target image.

4. Start the Configuration Tool again.

5. Navigate to the Runtime Environments (RTEs) panel:
   a. On the Main Menu panel select option 3 (Configure products).
      The Configure Products panel opens.
   b. Select option 1 (Select product to configure).
      The Product Selection Menu panel opens.
   c. Select a product that is configured in the runtime environment that you want to replicate. The Runtime Environments (RTEs) panel opens.

6. Generate the sample transport JCL:
   a. Type Z next to the runtime environment you want to transport and press Enter.
      The RTE Utility Menu panel opens.
   b. Select 5 (Generate sample transport JCL) and press Enter.
      This action causes several sample transport jobs to be generated in the &rhilev.&rte.RKANSAMU library. For more information about these jobs, see "Step 4. Generate sample transport JCL" on page 130.

7. Use sample job XDFDMP03 on the first image to dump the runtime environment configuration batch jobs.

8. Use sample job XDFRST03 on the target image to restore the batch jobs on the target image.

9. On the target image, submit the batch jobs in the order listed in the Jobs Sorted By Generation Sequence section of the batch mode job report.
   You can submit each job manually or run a product-provided CLIST to submit the Configuration Tool jobs on the target image. To use the CLIST to submit the jobs:
   a. Verify that the target libraries that were installed by SMP/E are available on the image where the CLIST will be executed.
   b. Edit the member named SUB#ssss in the &shilev.INSTJOBS data set, where ssss identifies the JCL suffix for the new runtime environment.
   c. Run the CLIST to submit the Configuration Tool jobs that will create the runtime environment.
   d. Exit the &shilev.INSTJOBS data set so that CICATB can run.
      You have successfully created a new runtime environment on the target image.

10. Perform any manual configuration steps on the target image, such as:
   • Copying procedures to PROCLIB
   • Copying VTAM definitions to VTAMLST
   • APF-authorizing libraries
      See "Step 6. Complete the configuration of a new runtime environment" on page 137 for more information.
Scenario D: Transporting runtime environment batch mode parameters from one z/OS image to another

Follow these steps to transport runtime environment batch mode parameters from one z/OS image to another.

1. Start the Configuration Tool on your first image using this command:
   
   ```sh
   EX '&shilev.INSTLIB'
   ```

2. Navigate to the Runtime Environments (RTEs) panel:
   a. On the Main Menu panel select option 3 (Configure products).
   b. Select option 1 (Select product to configure).

3. Generate the sample transport JCL:
   a. Type Z next to the runtime environment you want to transport and press Enter.
   b. Select 5 (Generate sample transport JCL) and press Enter.

4. On the master image, use sample job XDFDMP03 to dump the batch jobs.

5. On the target image, use XDFRST03 to restore the batch jobs.

6. On the target image, create the Configuration Tool batch mode job:
   a. Start the Configuration Tool by entering the following command in a TSO command panel:

   ```sh
   ===> EXEC '&shilev.INSTLIB'
   ```

   b. From the Main Menu, select Option 1 (Set up work environment) and press Enter.

   c. Select option 3 (Create batch mode job).

7. Exit the Configuration Tool.

8. Perform a scan on your runtime environment parameters:
   a. Edit CICATB, updating the BATCHLIB and BATCHMEM parameters as necessary, and setting the SUBMIT parameter to SCAN.
   b. Submit the CICATB job to scan your runtime environment parameters.
   c. Verify that the job completes successfully, review the parameter report, correct any errors in the parameter member, and repeat the scan until a clean report is generated.

9. Create the runtime environment on the target image:
   a. Edit CICATB again, setting the SUBMIT parameter to YES.
   b. Submit the CICATB job to create the runtime environment.
   c. Verify that the job completes successfully.

Perform any manual configuration steps on the target image, such as:
- Copying procedures to PROCLIB
- Copying VTAM definitions to VTAMLST
- APF-authorizing libraries
**Step 6. Complete the configuration of a new runtime environment**

Batch mode configuration requires manual steps after the configuration is complete. These steps are listed in a Complete the Configuration file, which can be accessed only from within the Configuration Tool. This file is created specifically for each configuration and its content depends on the configured components.

To generate the file:

1. From the Configuration Tool Main Menu ([Figure 27 on page 79]), enter 3 (Configure products).
2. On the Product Selection Menu, select any product.
3. On the Runtime Environments (RTEs) panel, enter Z (Utilities) next to the runtime environment in which you want to complete product configuration.
4. On the RTE Utility Menu ([Figure 52 on page 175]), enter 9 (Create Consolidated Complete the Configuration instructions).
   
   This option creates JCL containing a consolidated list of Complete the Configuration instructions for all products configured in the runtime environment. The list is stored in the &shilev:INSTJOBS data set, in a member named DFI@ssss (where ssss is the unique JCL suffix for the runtime environment).
5. Perform the tasks listed in the Complete the Configuration job. The list is generated dynamically and tailored specifically to your configuration.

   The most common tasks for configuring a monitoring server are covered in Chapter 6, “Completing the configuration,” on page 103. For detailed information about completing the configuration of monitoring agents, see the configuration documentation for each monitoring agent in the Tivoli Monitoring and OMEGAMON XE Information Center at [http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?toc=](http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?toc=)

---

**Tips**

- You do not have to configure the monitoring server in a runtime environment more than once, even though the monitoring server shows up on the Product Component Selection Menu for every monitoring agent.
- The Complete the Configuration list for every monitoring agent on z/OS includes the task of registering the monitoring agent with the monitoring server in its runtime environment. See *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS*.

See the “Complete the configuration” chapter in the configuration guides for each product installed in the runtime environment for more information about required post-Configuration Tool configuration steps specific to that product.
Chapter 8. Configuration Tool (ICAT) deployment scenarios

This chapter contains a series of thumbnail scenarios that illustrate various methods for using the Configuration Tool to replicate a configuration of Tivoli Management Services components and monitoring agents across an enterprise. These scenarios assume that you are using system variables (IBM symbolics) and consistent naming conventions for data set names, started tasks, and VTAM applids on all your LPARs. For detailed configuration instructions, see Part 2, “Configuring components on z/OS,” on page 53. For information about using system variables, see Appendix A, “Enabling system variable support,” on page 165.

The following sections cover several enterprise-wide deployment scenarios:

- “Scenario 1: Standard replication method”
- “Scenario 2: Using common RKANPARU and RKANCMDU libraries” on page 141
- “Scenario 3: Using batch mode to replicate a runtime environment to another LPAR” on page 143
- “Scenario 4: Using batch auto-submit to replicate a runtime environment to another LPAR” on page 144
- “Scenario 5: Using batch mode to replicate a runtime environment on the same LPAR” on page 146
- “Scenario 6: Using interactive copy to replicate a runtime environment” on page 147
- “Scenario 7: Copying configuration values from one instance to another of the Configuration Tool” on page 147
- “Scenario 8: Configuring a remote monitoring server and monitoring agents in batch mode” on page 148

Scenario 1: Standard replication method

This scenario describes a standard method for replicating a configuration of products across a z/OS enterprise. You install and configure all products on your main LPAR, and then copy items from the main LPAR to other LPARs throughout your enterprise.

Scenario 1 has the following advantages:

- You install and configure only one instance of the Configuration Tool.
- You configure a maximum of three unique runtime environments.
- Products use symbolic values defined on the LPAR they run on.
- Replication across the enterprise is quick.
- Runtime environments are consistent.
- You save storage space.

The disadvantage of this scenario is that you cannot use the Configuration Tool to update the runtime environments on other LPARs.

Table 12. Thumbnail configuration scenario 1: standard replication method

<table>
<thead>
<tr>
<th>CPC A</th>
<th>CPC B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared DASD in CPC A</td>
<td>Shared DASD in CPC B</td>
</tr>
<tr>
<td>1 Configuration Tool and</td>
<td>1 copy of base runtime environment from LPAR A1</td>
</tr>
<tr>
<td>1 SMP/E environment</td>
<td>3 copies of sharing runtime environment A3_remote</td>
</tr>
<tr>
<td>1 base runtime environment</td>
<td>on LPAR A4</td>
</tr>
<tr>
<td>2 sharing runtime environments:</td>
<td></td>
</tr>
<tr>
<td>A1_hub and A3_remote</td>
<td></td>
</tr>
<tr>
<td>1 copy of sharing runtime environment A3_remote on LPAR A4</td>
<td></td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 2008, 2012
Table 12. Thumbnail configuration scenario 1: standard replication method (continued)

<table>
<thead>
<tr>
<th>Configuration Tool, target libraries that were installed by SMP/E</th>
<th>Base runtime environment</th>
<th>Sharing runtime environment</th>
<th>RKANPARU member</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool, target libraries that were installed by SMP/E</td>
<td>Copy of base runtime environment from A1, renamed to B1</td>
<td>Copy of sharing runtime environment from A3_remote, renamed to A4</td>
<td>Copy of sharing runtime environment from A3_remote, renamed to A1_hub</td>
</tr>
<tr>
<td>Base runtime environment</td>
<td></td>
<td>Sharing runtime environment A1_hub</td>
<td>RKANPARU member A1_hub</td>
</tr>
<tr>
<td>Sharing runtime environment A1_hub</td>
<td></td>
<td>Sharing runtime environment A3_remote</td>
<td>RKANPARU member A3_remote</td>
</tr>
<tr>
<td>RKANPARU member A1_hub</td>
<td></td>
<td>RKANPARU member A4</td>
<td>RKANPARU member A4</td>
</tr>
</tbody>
</table>

To use scenario 1, perform the following steps:

1. Install all your products on central processor complex (CPC) A, LPAR A1 (main).
   This sets up one Configuration Tool and one SMP/E environment to support all LPARs. See the
   program directories for your products for detailed installation instructions.
2. Create a base runtime environment on LPAR A1 (main).
3. Create a sharing-with-base runtime environment on LPAR A1 (main), to contain the hub monitoring
   server and a set of monitoring agents. Be sure to enable system variables. This runtime environment
   becomes your template for hub sharing-with-base runtime environments on other LPARs.
4. Configure the hub monitoring server and monitoring agents.
5. Create a second sharing-with-base runtime environment on LPAR A3. Be sure to enable system
   variables. This runtime environment becomes your template for remote sharing-with-base runtime
   environments on other LPARs.
6. In this runtime environment, configure a remote monitoring server to report to the hub in LPAR A1,
   and a set of monitoring agents to report to the remote monitoring server.
7. Create the system variable parameter member. See "Create the system variable parameter member"
   on page 169.
8. (Optional) Create a single VTAM major node. See "Create one VTAM major node for all products in
   the runtime environment" on page 170.
9. Copy the base runtime libraries from LPAR A1 (main) to LPAR B1 (target), by using any
    copy-and-rename method or by using the runtime environment transport batch jobs provided by the
    Configuration Tool. Keep the same runtime library names, if possible. See "Step 4. Generate sample
    transport JCL" on page 130.
10. Copy and rename the remote sharing-with-base runtime libraries from LPAR A3 to LPAR B1, by using
    any copy-and-rename method or by using the runtime environment transport batch jobs provided by the
    Configuration Tool. As the files are copied, rename them to match the LPAR name.
11. In each &rhilev.&tte.RKANPARU library, create a new system variable member that matches the
    name of the LPAR, and copy into it the contents of the system variable member you created earlier.
    In the new member, change LPAR-specific values as required.
12. Repeat the preceding three steps for all other LPARs in your enterprise.
13. Complete the configuration on each system. At a minimum you must complete these steps:
   a. Copy the started tasks and VTAM major nodes to the system libraries.
   b. APF-authorize the data sets on the appropriate LPARs.
14. Start each monitoring server and monitoring agent.

**Scenario 2: Using common RKANPARU and RKANCMDU libraries**

This scenario describes another method for replicating a configuration of products across your z/OS enterprise. In the scenario, you create a new common RKANPARU library and a new common RKANCMDU library to contain the configuration values required for runtime environments in all of your LPARs. You install and configure all products on your main LPAR, and then copy items from the main LPAR to other LPARs throughout your enterprise.

Scenario 2 has the following advantages:
- You use one common set of RKANPARU and RKANCMDU libraries for all LPARs.
- You install and configure only one instance of the Configuration Tool.
- You configure a maximum of three unique runtime environments.
- Runtime environments are copied to other LPARs.
- Products use symbolic values defined on the LPAR they run on.
- Replication across the enterprise is quick.
- Runtime environments are consistent.
- You save storage space.

The disadvantage of this scenario is that you cannot use the Configuration Tool to update the runtime environments on other LPARs.

*Table 13. Thumbnail configuration scenario 2: Using common RKANPARU and RKANCMDU libraries*

<table>
<thead>
<tr>
<th>CPC A</th>
<th>CPC B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared DASD in CPC A</td>
<td>Shared DASD in CPC B</td>
</tr>
<tr>
<td>1 Configuration Tool and 1 SMP/E environment</td>
<td>1 copy of base runtime environment from LPAR A1</td>
</tr>
<tr>
<td>1 base runtime environment</td>
<td>3 copies of sharing runtime environment A3_remote</td>
</tr>
<tr>
<td>2 sharing runtime environments: A1_hub and A3_remote</td>
<td>1 common RKANPARU data set</td>
</tr>
<tr>
<td>1 copy of sharing runtime environment A3_remote on LPAR A4</td>
<td>1 common RKANCMDU data set</td>
</tr>
<tr>
<td>1 common RKANPARU data set</td>
<td>1 common RKANPARU data set</td>
</tr>
<tr>
<td>1 common RKANCMDU data set</td>
<td>1 common RKANCMDU data set</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LPAR A1 (main)</th>
<th>LPAR A3</th>
<th>LPAR A4</th>
<th>LPAR B1 (target)</th>
<th>LPAR B4</th>
<th>LPAR B7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Tool, target libraries that were installed by SMP/E</td>
<td>Copy of base runtime environment from A1, renamed to B1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To use scenario 2, perform the following steps:

1. Install all your products on CPC A, LPAR A1 (main).
   This sets up one Configuration Tool and one SMP/E environment to support all LPARs. See the
   program directories for your products for detailed installation instructions.

2. Create a base runtime environment on LPAR A1 (main).

3. Create a sharing-with-base runtime environment on LPAR A1 (main), to contain the hub monitoring
   server and a set of monitoring agents. Be sure to enable system variables.

4. Configure the hub monitoring server and monitoring agents in the sharing-with-base runtime
   environment. This runtime environment becomes your template for hub sharing-with-base runtime
   environments on other LPARs.

5. Create a second sharing-with-base runtime environment on LPAR A3. Be sure to enable system
   variables. This runtime environment becomes your template for remote sharing-with-base runtime
   environments on other LPARs.

6. In this runtime environment, configure a remote monitoring server to report to the hub in LPAR A1,
   and a set of monitoring agents to report to the remote monitoring server.

7. Create the system variable parameter member. See "Create the system variable parameter member"
   on page 169.

8. Create common RKANPARU and RKANCMDU data sets, and copy the corresponding template
   runtime environment data sets into these common data sets.

9. In the common RKANPARU, create additional system variable members as necessary.
   The names of these members must match the other LPAR names. For each member, change
   LPAR-specific values as necessary. Review cross-system values for components on different LPARs.

10. Edit RKANSAMU started tasks, changing all occurrences of the RKANPARU and RKANCMDU data
    set names to the common data set names.
    For example:
    
    `&rilev.&rte.RKANPARU` `common.RKANPARU` all
    `&rilev.&rte.RKANCMDU` `common.RKANCMDU` all

11. (Optional) Create a single VTAM major node. See "Create one VTAM major node for all products in
    the runtime environment" on page 170.

12. Copy the following data sets to LPAR B1 (target), by using any copy-and-rename method or by using
    the runtime environment transport batch jobs provided by the Configuration Tool:
      - Base runtime environment data sets
      - Common RKANPARU and RKANCMDU data sets
    Keep the same data set names, if possible.

13. Copy and rename the remote sharing-with-base runtime libraries from LPAR A1 to LPAR B1, by using
    any copy-and-rename method or by using the runtime environment transport batch jobs provided by
    the Configuration Tool. As the files are copied, rename them to match the LPAR name.

14. Repeat the preceding two steps for all other LPARs in your enterprise.

---

**Table 13. Thumbnail configuration scenario 2: Using common RKANPARU and RKANCMDU libraries (continued)**

<table>
<thead>
<tr>
<th>Sharing runtime environment A3_remote</th>
<th>Copy of sharing runtime environment A3_remote, renamed to A4</th>
<th>Copy of sharing runtime environment A3_remote, renamed to B1</th>
<th>Copy of sharing runtime environment A3_remote, renamed to B4</th>
<th>Copy of sharing runtime environment A3_remote, renamed to B7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common RKANPARU and RKANCMDU data sets with members A1, A3, A4, B1, B4, B7</td>
<td>Common RKANPARU and RKANCMDU data sets with members A1, A3, A4, B1, B4, B7</td>
<td>Common RKANPARU and RKANCMDU data sets with members A1, A3, A4, B1, B4, B7</td>
<td>Common RKANPARU and RKANCMDU data sets with members A1, A3, A4, B1, B4, B7</td>
<td>Common RKANPARU and RKANCMDU data sets with members A1, A3, A4, B1, B4, B7</td>
</tr>
</tbody>
</table>
15. Complete the configuration on each system. At a minimum you must complete these steps:
   a. Copy the started tasks and VTAM major nodes to the system libraries.
   b. APF-authorize the data sets on the appropriate LPARs.

16. Start each monitoring server and monitoring agent.

**Scenario 3: Using batch mode to replicate a runtime environment to another LPAR**

This scenario describes a method for copying a runtime environment to another LPAR, using batch mode in the Configuration Tool.

Scenario 3 has the following advantages:
- You have less work to do in interactive mode in the Configuration Tool.
- You create each runtime environment after the first with a single batch job.
- Detailed reports help you troubleshoot and edit the batch job.

The only disadvantage is that you must install and set up an instance of the Configuration Tool on each LPAR.

**Table 14. Thumbnail configuration scenario 3: Using batch mode to replicate a runtime environment to another LPAR**

<table>
<thead>
<tr>
<th></th>
<th>CPC A</th>
<th>CPC B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared DASD</td>
<td>Shared DASD in CPC A</td>
<td>Shared DASD in CPC B</td>
</tr>
<tr>
<td></td>
<td>1 Configuration Tool</td>
<td>1 Configuration Tool</td>
</tr>
<tr>
<td></td>
<td>1 SMP/E environment</td>
<td>1 copy of target libraries from SMP/E</td>
</tr>
<tr>
<td></td>
<td>1 base runtime environment</td>
<td>1 copy of base runtime environment</td>
</tr>
<tr>
<td></td>
<td>3 unique sharing runtime environments</td>
<td>3 unique sharing runtime environments</td>
</tr>
<tr>
<td>LPAR A1 (main)</td>
<td>Configuration Tool, target libraries that were installed by SMP/E</td>
<td>LPAR B1 (target)</td>
</tr>
<tr>
<td></td>
<td>Base runtime environment</td>
<td>Configuration Tool, target libraries copied from A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Base runtime environment copied from A1 and renamed to B1</td>
</tr>
<tr>
<td></td>
<td>Sharing runtime environment A1, created in interactive mode</td>
<td>Sharing runtime environment A1, created in batch mode</td>
</tr>
<tr>
<td></td>
<td>Sharing runtime environment A3, created in batch mode</td>
<td>Sharing runtime environment A3, created in batch mode</td>
</tr>
<tr>
<td></td>
<td>Sharing runtime environment A4, created in batch mode</td>
<td>Sharing runtime environment A4, created in batch mode</td>
</tr>
<tr>
<td></td>
<td>Sharing runtime environment B1, created in batch mode</td>
<td>Sharing runtime environment B1, created in batch mode</td>
</tr>
<tr>
<td></td>
<td>Sharing runtime environment B4, created in batch mode</td>
<td>Sharing runtime environment B4, created in batch mode</td>
</tr>
<tr>
<td></td>
<td>Sharing runtime environment B7, created in batch mode</td>
<td>Sharing runtime environment B7, created in batch mode</td>
</tr>
</tbody>
</table>

To use scenario 3, perform the following steps:
1. Install all your products on CPC A, LPAR A1 (main). See the program directories for your products for detailed installation instructions.
2. Copy the Configuration Tool libraries and the target libraries that were installed by SMP/E to CPC B, by using any copy-and-rename method or by using the runtime environment transport batch jobs provided by the Configuration Tool.
3. Create a base runtime environment on LPAR A1 (main).
See "Step 2. Create and clone a batch mode parameter member" on page 125 for detailed procedure.

6. Create a new batch parameter member for each of the other LPARs (A3, A4, B1, B4, and B7, by copying the contents of the sharing-with-base runtime environment A1 parameter member and renaming it to the name of the target runtime environment.

7. In each new batch parameter member, review and edit the values.

8. Submit the Configuration Tool batch job on CPC A for each LPAR in CPC A (A3 and A4). This creates the required runtime environments for CPC A.

9. Copy the batch parameter members for each LPAR in CPC B (B1, B4, B7) to the Configuration Tool on CPC B.

10. Submit the Configuration Tool batch job on CPC B for each LPAR in CPC B (B1, B4, B7). This creates the required runtime environments for CPC B.

11. Copy the base runtime environment A1 data sets to LPAR B1. Keep the same data set names, if possible.

12. Complete the configuration on each system. At a minimum you must complete these steps:
   a. Copy the started tasks and VTAM major nodes to the system libraries.
   b. APF-authorize the data sets on the appropriate LPARs.

13. Start each monitoring server and monitoring agent.

**Scenario 4: Using batch auto-submit to replicate a runtime environment to another LPAR**

This scenario describes a method for copying a runtime environment to another LPAR by using the auto-submit feature of batch mode processing in the Configuration Tool. This scenario is similar to scenario 3, except that the new runtime environments are created by JCL that is built on the main LPAR, using Configuration Tool batch mode processing. You copy the JCL to the target LPAR and run a product-provided CLIST to submit the necessary jobs.

Scenario 4 has the following advantages:
- You install and configure only one instance of the Configuration Tool.
- The Configuration Tool contains all runtime environment configuration values.
- All JCL is built on the main LPAR.
- You have less work to do in interactive mode in the Configuration Tool.
- You create each runtime environment after the first with a single batch job.
- Detailed reports help you troubleshoot and edit the batch job.

This scenario has no disadvantages.

*Table 15. Thumbnail configuration scenario 4: Using batch auto-submit to replicate a runtime environment to another LPAR*

<table>
<thead>
<tr>
<th>CPC A</th>
<th>CPC B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared DASD in CPC A</td>
<td>Shared DASD in CPC B</td>
</tr>
<tr>
<td>1 Configuration Tool</td>
<td>1 copy of target libraries from SMP/E</td>
</tr>
<tr>
<td>1 SMP/E environment</td>
<td>1 copy of base runtime environment</td>
</tr>
<tr>
<td>1 base runtime environment</td>
<td>3 unique sharing runtime environments</td>
</tr>
<tr>
<td>3 unique sharing runtime environments</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LPAR A1 (main)</th>
<th>LPAR A3</th>
<th>LPAR A4</th>
<th>LPAR B1 (target)</th>
<th>LPAR B4</th>
<th>LPAR B7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Tool, target libraries that were installed by SMP/E</td>
<td>Target libraries copied from A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

144 IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Common Planning and Configuration Guide
Table 15. Thumbnail configuration scenario 4: Using batch auto-submit to replicate a runtime environment to another LPAR (continued)

<table>
<thead>
<tr>
<th>Base runtime environment</th>
<th>Sharing runtime environment A1, created in interactive mode</th>
<th>JCL generated in batch mode for creating runtime environments A3, A4, B1, B4, B7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base runtime environment copied from A1 and renamed to B1</td>
<td>Sharing runtime environment A3, created in batch mode</td>
<td>JCL generated in batch mode for creating runtime environment A3</td>
</tr>
<tr>
<td>Base runtime environment copied from A1 and renamed to B1</td>
<td>Sharing runtime environment A4, created in batch mode</td>
<td>JCL generated in batch mode for creating runtime environment A4</td>
</tr>
<tr>
<td>Base runtime environment copied from A1 and renamed to B1</td>
<td>Sharing runtime environment B1, created in batch mode</td>
<td>JCL generated in batch mode for creating runtime environment B1</td>
</tr>
<tr>
<td>Base runtime environment copied from A1 and renamed to B1</td>
<td>Sharing runtime environment B4, created in batch mode</td>
<td>JCL generated in batch mode for creating runtime environment B4</td>
</tr>
<tr>
<td>Base runtime environment copied from A1 and renamed to B1</td>
<td>Sharing runtime environment B7, created in batch mode</td>
<td>JCL generated in batch mode for creating runtime environment B7</td>
</tr>
</tbody>
</table>

To use scenario 4, perform the following steps:

1. Install all your products on CPC A, LPAR A1 (main). See the program directories for your products for detailed installation instructions.

2. Copy the target libraries that were installed by SMP/E to CPC B, by using any copy-and-rename method or by using the runtime environment transport batch jobs provided by the Configuration Tool.

3. Create a base runtime environment on LPAR A1 (main).


   
   See “Step 2. Create and clone a batch mode parameter member” on page 125 for detailed procedure.

6. Create a new batch parameter member for each of the other LPARs (A3, A4, B1, B4, and B7, by copying the contents of the sharing-with-base runtime environment A1 parameter member and renaming it to the name of the target runtime environment.

7. In each new batch parameter member, review and edit the values.

8. On LPAR A1, submit the Configuration Tool batch jobs for all LPARs in CPC A (A3 and A4) and for all LPARs in CPC B (B1, B4, B7). Set the SUBMIT parameter to YES in the batch job options for LPARs A3 and A4, and to NO for all CPC B LPARs.

   These batch jobs create the required runtime environments for CPC A and generate the JCL necessary for creating the runtime environments on CPC B.

9. Copy the JCL members from &shilev:INSTJOBS to each LPAR in CPC B.

   The names of these members end in the 4-character suffix that you specified in the batch parameter member.

10. On CPC B, run the product-provided CLIST named SUB#ssss to submit the batch jobs required to create runtime environments B1, B4, and B7.

11. Copy the base runtime environment A1 data sets to LPAR B1. Keep the same data set names, if possible.

12. Complete the configuration on each system. At a minimum you must complete these steps:
   
   a. Copy the started tasks and VTAM major nodes to the system libraries.

   b. APF-authorize the data sets on the appropriate LPARs.

13. Start each monitoring server and monitoring agent.
Scenario 5: Using batch mode to replicate a runtime environment on the same LPAR

This scenario describes a method for replicating a runtime environment on the same LPAR, using batch mode processing in the Configuration Tool. You install and configure all products on your main LPAR, and then copy the runtime environments to the appropriate LPARs.

Scenario 5 has the following advantages:
- You install and configure only one instance of the Configuration Tool.
- You have less work to do in interactive mode in the Configuration Tool.
- You create each runtime environment after the first with a single batch job.
- Detailed reports help you troubleshoot and edit the batch job.
- Making copies of a runtime environment without moving to a different LPAR is the easiest way to replicate a runtime environment in a large enterprise.

The disadvantage is that duplicated runtime environments temporarily use more storage space.

Table 16. Thumbnail configuration scenario 5: Using batch mode to replicate a runtime environment on the same LPAR

<table>
<thead>
<tr>
<th>CPC A</th>
<th>CPC B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared DASD in CPC A</td>
<td>Shared DASD in CPC B</td>
</tr>
<tr>
<td>1 Configuration Tool</td>
<td></td>
</tr>
<tr>
<td>1 SMP/E environment</td>
<td></td>
</tr>
<tr>
<td>1 base runtime environment</td>
<td>1 copy of base runtime environment</td>
</tr>
<tr>
<td>3 unique sharing runtime environments</td>
<td>3 copies of sharing runtime environments</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LPAR A1 (main)</th>
<th>LPAR A3</th>
<th>LPAR A4</th>
<th>LPAR B1 (target)</th>
<th>LPAR B4</th>
<th>LPAR B7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Tool, target libraries that were installed by SMP/E</td>
<td></td>
<td></td>
<td>Base runtime environment copied from A1 and renamed to B1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base runtime environment</td>
<td></td>
<td></td>
<td>Sharing runtime environment A1, created in interactive mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sharing runtime environment A1, created in interactive mode</td>
<td>Sharing runtime environment A3, copied from A1</td>
<td>Sharing runtime environment A4, copied from A1</td>
<td>Sharing runtime environment B1, copied from A1</td>
<td>Sharing runtime environment B4, copied from A1</td>
<td>Sharing runtime environment B7, copied from A1</td>
</tr>
<tr>
<td>Sharing runtime environments A3, A4, B1, B4, and B7, created in batch mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To use scenario 5, perform the following steps:
1. Install all your products on CPC A, LPAR A1 (main).
   This sets up one Configuration Tool and one SMP/E environment to support all LPARs. See the program directories for your products for detailed installation instructions.
2. Create a base runtime environment on LPAR A1.
   See "Step 2. Create and clone a batch mode parameter member" on page 125 for detailed procedure.

5. Create a new batch parameter member for each of the other LPARs (A3, A4, B1, B4, and B7), by copying the contents of the sharing-with-base runtime environment A1 parameter member and renaming it to the name of the target runtime environment.

6. In each new batch parameter member, review and edit the values.

7. Submit the Configuration Tool batch job for each LPAR, to create the required runtime environments.

8. Copy the base runtime environment A1 data sets to LPAR B1, by using any copy-and-rename method or by using the runtime environment transport batch jobs provided by the Configuration Tool. Keep the same data set names, if possible.

9. Copy the appropriate sharing-with-base runtime environment to each LPAR, by using any copy-and-rename method or by using the runtime environment transport batch jobs provided by the Configuration Tool. As the files are copied, rename them to match the destination LPAR name.

10. Complete the configuration on each system. At a minimum you must complete these steps:
     a. Copy the started tasks and VTAM major nodes to the system libraries.
     b. APF-authorize the data sets on the appropriate LPARs.

11. Start each monitoring server and monitoring agent.

### Scenario 6: Using interactive copy to replicate a runtime environment

This scenario describes an interactive method for creating a runtime environment based on values specified in an existing runtime environment. The advantage of this method is that it saves duplication of effort in interactive mode. The disadvantage is that you must navigate through all the configuration panels in the Configuration Tool to review and edit values for the new runtime environment and to submit the configuration jobs.

**Table 17. Thumbnail configuration scenario 6: Using interactive copy to replicate a runtime environment**

<table>
<thead>
<tr>
<th>LPAR A1 (main)</th>
<th>LPAR A3 (target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Tool, target libraries that were installed by SMP/E</td>
<td>Runtime environment A3</td>
</tr>
<tr>
<td>Runtime environment A1</td>
<td></td>
</tr>
</tbody>
</table>

To use scenario 6, perform the following steps:

2. Create runtime environment A3, specifying the name of runtime environment A1 in this field on the Add Runtime Environment panel:
   Copy configuration values from RTE =>>
3. Complete all configuration steps and jobs for runtime environment A3.

### Scenario 7: Copying configuration values from one instance to another of the Configuration Tool

This scenario describes a method for copying configuration values, one runtime environment at a time, from one instance to another of the Configuration Tool. The advantage of this method is that it reduces duplication of effort in configuring products. The disadvantage is that you must install and set up more than one instance of the Configuration Tool.
Table 18. Thumbnail configuration scenario 7: Copying configuration values from one instance to another of the Configuration Tool

<table>
<thead>
<tr>
<th>Configuration Tool #1</th>
<th>Configuration Tool #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch parameter member for runtime environment A</td>
<td>Batch mode job using the parameter member from runtime environment A</td>
</tr>
<tr>
<td></td>
<td>ISPF tables storing the runtime environment values</td>
</tr>
</tbody>
</table>

To use scenario 7, perform the following steps:
1. In interactive mode in Configuration Tool #1, create and configure runtime environment A.
2. Create a batch parameter member for runtime environment A.
3. In Configuration Tool #2, run batch mode job CICATB, by using the batch parameter member from runtime environment A. Specify SUBMIT(NO) if you want to populate the Configuration Tool ISPF tables and generate JCL in the &shilev:INSTJOBS data set without submitting the jobs to create and configure a second runtime environment.

**Scenario 8: Configuring a remote monitoring server and monitoring agents in batch mode**

This scenario shows how to use batch mode processing to create a new runtime environment that contains a remote z/OS monitoring server and monitoring agents. When you use batch mode, you create new runtime environments by submitting batch jobs, rather than by configuring them panel by panel. You use the configuration values from an existing runtime environment as the model for the new runtime environment and modify the values as required to define the new environment.

*Figure 42. Batch scenario configuration: z/OS remote monitoring server reporting to z/OS hub*
In the scenario shown in Figure 42 on page 148, an existing runtime environment (here called SYSA) is used as the model for the new runtime environment. The SYSA runtime environment has the following characteristics:

- It was created by using the Configuration Tool ISPF screens.
- It contains a hub monitoring server and several monitoring agents.
- It uses system variables.
- The sysplex proxy is defined to run in the hub. (The sysplex proxy is a component of OMEGAMON XE on z/OS, which is one of the monitoring agents in the runtime environment.)
- Several OMEGAMON XE products are configured in the runtime environment. All agents defined in the runtime environment run in stand-alone address spaces, with the exception of OMEGAMON XE on z/OS and OMEGAMON XE for Storage on z/OS, which must run in the monitoring server address space.

The new runtime environment created with batch mode (here called SYSB) has the following characteristics:

- It contains a remote monitoring server, reporting to the hub in SYSA.
- It uses system variables.
- It has the same agents as SYSA, but the agents connect to the monitoring server in SYSB.
- SYSB is in the same sysplex as SYSA.

For examples of other configuration scenarios that use batch mode processing, see Chapter 8, “Configuration Tool (ICAT) deployment scenarios,” on page 139. See Chapter 7, “Using the Configuration Tool (ICAT) batch mode to replicate a configured environment,” on page 121 for more information on batch mode processing.

**Configuration steps**

Complete the following steps in order:

1. **Step 1. Create the batch mode job (CICATB)**
2. **Step 2. Clone the runtime environment batch parameter member** on page 150
3. **Step 3. Review and update the parameter member for the new runtime environment** on page 153
4. **Step 4. Run the CICATB job to validate the changes** on page 156
5. **Step 5. Create and (optionally) execute the batch jobs to create the runtime environment** on page 157
6. **Step 6. Customize and complete the configuration** on page 162

**Step 1. Create the batch mode job (CICATB)**

The CICATB job generates the configuration jobs required to build, configure and load the products configured in a runtime environment. CICATB can also be used to validate the configuration values used by the configuration jobs. CICATB does not exist by default. You generate it by using the Configuration Tool.

Complete the following steps to create the job:

1. Start the Configuration Tool by entering the following command in a TSO command panel:
   ```
   ====> EXEC '&shilev.INSTLIB'
   ```
2. From the Main Menu, select Option 1 (Set up work environment) and press Enter.
   The Set up work environment panel is displayed.
3. Select option 3 (Create batch mode job).
4. Wait until you see the **CICATB JOB CREATED** message in the upper-right corner of the panel.

The **SYSTSIN DD** statement in the JCL contains several parameters that you must customize before you submit the job.

**BATCHLIB**

The library that contains the batch runtime environment parameter member. The default is

\&shilev.INSTJOBS.

**BATCHMEM**

The member name of the batch runtime environment parameter member (the name must be the same as the name of the runtime environment you intend to create). You create this member in the following step.

**SUBMIT**

The type of processing. You specify different dispositions depending on the step you are performing and the requirements of the environment.

- **YES** Creates and submits the runtime environment configuration jobs that create the runtime environment, if the jobs do not already exist. If the jobs already exist, such as when you are reconfiguring a runtime environment, the jobs are regenerated but are not submitted for execution.
- **RERUN** Creates and submits the runtime environment configuration jobs, regardless of whether the jobs already exist.
- **NO** Creates but does not submit the runtime environment configuration jobs.
- **SCAN** Validates the input parameter member and produces a report.
- **SCAND** Scans the input parameter member and produces a detailed data dictionary of all the runtime environment parameters.

**Important**

When you create the CICATB batch job member in the \&shilev.INSTJOBS data set, the Configuration Tool also creates the batch jobcard ISPF table KCITPIG1 in the \&shilev.INSTDATA data set. The contents of the KCITPIG1 table are used as the standard jobcard in batch configuration jobs. If you modify the jobcard on the Specify JCL Options panel (Figure 29 on page 80) or if the ISPF environment changes in some other manner, you must recreate the CICATB batch job to refresh the KCITPIG1 jobcard table before running CICATB.

---

**Step 2. Clone the runtime environment batch parameter member**

A batch mode parameter member contains configuration values for all the products in an existing runtime environment. The member created, or recreated, every time the runtime environment Load job is run. It can also be generated by using the “Create/Clone batch mode parameters” utility. To create a new runtime environment using batch processing, you clone a copy of the member for an existing runtime environment and modify the values as required for the target image. The name you give to the cloned member becomes the mid-level qualifier for the new runtime environment you create by using the member.
Take the following steps to clone a parameter member:

1. Start the Configuration Tool by entering the following command in a TSO command panel:

   ```
   ===> EXEC '&\shilev.INSTLIB'
   ```

2. On the Main Menu panel, select option 3 (Configure products).

   The Configure Products panel is displayed (as shown in Figure 31 on page 84).

3. Select 1 (Select product to configure) and press Enter. The Product Selection Menu is displayed (as shown in Figure 32 on page 84).

4. Select any product to bring up the Runtime Environments (RTEs) panel (Figure 43).

   The existing runtime environment is named SYSA. It is a sharing runtime environment that uses the SMP/E libraries and contains a hub monitoring server.

5. Enter Z (Utilities) next to SYSA and press Enter to display the RTE Utility Menu (Figure 44).

6. Select option 1 (Create/Clone batch mode parameters) and press Enter. The Create/Clone Parameter Deck Menu is displayed (Figure 45 on page 152).
7. Because this scenario involves changing a hub monitoring server in the model runtime environment into a remote monitoring server in the replicated runtime environment, select option 3 (Create batch mode parameter deck, with modifications) and press Enter. This option also allows you the ability to exclude products defined in the model environment from the target environment.

The Create/Clone Batch Mode Parameters panel is displayed (Figure 46).

Figure 45. Create/Clone Parameter Deck Menu panel

KCIMBATO ------- CREATE/CLONE PARAMETER DECK MENU / RTE: SYSA -----------
OPTION ===>
Specify the number of the desired action.
  1 Create batch mode parameter deck.
  2 Clone batch mode parameter deck, unmodified.
  3 Clone batch mode parameter deck, with modifications.
F1=Help F3=Back

Figure 46. Create/Clone Batch Mode Parameters panel

The first section of the panel, Create Parameter Deck Member, contains default values for the parameter member with values from the existing runtime environment. If a parameter deck does not already exist, the Configuration Tool will create it. By default, the name of the parameter member is the name of the runtime environment and it is created in the &shilev.INSTJOBS library. If you want to examine the parameter member after it is created, leave Y as the value for Edit after create. Otherwise, type N.

For Clone Parameter Deck Member, specify appropriate values for the following fields:

Parameter library
The library where the clone parameter member is to be created. By default, the library is &shilev.INSTJOBS. You can change the library name, but the library that you specify must already exist. If the library does not exist, the Configuration Tool does not create the parameter member and display the message “LIBRARY ERROR” in the upper right corner of the panel.

Clone member name
The name of the clone member. This name becomes the mid-level qualifier for the new runtime environment. For this scenario, the name is SYSB.

Edit after create
Specify Y to have the Configuration Tool open the member in ISPF edit mode so you can review or edit the values. If you want to edit the member later, specify N.
8. Press Enter.

If the input parameter deck member does not already exist, the following message is displayed:

The parameter deck member SYSA does not exist.
Press Enter to continue and create RTE parameters
or F3 to cancel.

Press Enter to create the parameter deck.

After parameter member for the model runtime environment is created, the following message is displayed:

BATCH PARMS CREATED

9. If necessary, press Enter again to clone the parameter deck.

The following message is displayed:

```
------- Change RTE JCL suffix panel -------

The cloned batch parameter deck includes
a value for the RTE JCL suffix.

Every RTE should have a unique one to
four character RTE_JCL_SUFF value.

The RTE_JCL_SUFF value for the original
RTE SYSA = SYSA

Specify a new unique RTE_JCL_SUFF for
RTE SYSB ===> __

ENTER=Next  F3=Abort Cloning
```

10. Type the JCL suffix for the new runtime environment and press Enter.

If you specified Y for Edit after create, the clone member is displayed.

Step 3. Review and update the parameter member for the new runtime environment

If you specified Y for Edit after create, the parameter member is already displayed in ISPF edit mode. If you did not specify Y, locate the member in the specified library and open it. You see something similar to Figure 47 on page 154
The contents of the parameter member are structured as follows:

- Variable names and values are defined within sections.
- The first variable name section is bounded by RTE$vvv BEGIN and RTE$vvv END lines, where vvv is the version of the Configuration Tool. Between these lines are variable names and values for the runtime environment variables.
- Following the runtime environment variable name section are sections for all products that are configured in the runtime environment. These sections are bounded by Kpp$vvv BEGIN and Kpp$vvv END lines, where pp is the 2-character product code, and vvv is the product version.

For example, the variable name section containing the OMEGAMON XE on z/OS V4.2.0 (product code M5) variable definitions begins with

```
KM5$420 BEGIN *---------- OMEGAMON XE on z/OS V420 -------------*
```

- Comment lines begin with an asterisk.
- The first value on a line is the variable name. The second value is the variable value. Each variable value matches a value supplied on a configuration panel for the runtime environment or for the product. In some cases, the value is enclosed by double quotation marks.

Table 19 lists the variables that you must update to create a runtime environment with the desired characteristics.

<table>
<thead>
<tr>
<th>Required actions</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the JCL suffix for jobs generated for this runtime environment.</td>
<td>RTE_JCL_SUFF</td>
</tr>
<tr>
<td>You must change the suffix because the CICATB job does not submit the jobs it creates if they already exist in INSTJOBS.</td>
<td></td>
</tr>
</tbody>
</table>
Table 19. Actions required to update the runtime environment parameters (continued)

<table>
<thead>
<tr>
<th>Required actions</th>
<th>Variables</th>
</tr>
</thead>
</table>
| Set the prefix used in generating started task names. | RTE_STC_PREF  
*Note:* Some variables, most ending in _STC, already have started task names. Update the existing names with the new prefix. |
| Change the prefix used in generating VTAM APPLIDs | RTE_VTAM_APPL_PREF  
*Note:* Other variables, containing the string _APPL_, have VTAM APPLIDs. Search the member for instances of the VTAM APPLID prefix, and update the prefix wherever it occurs. |
| Set the default TCP/IP communications values:  
- Host name  
- TCP/IP started task name  
- IP port for communications  
*Note:* If the remote monitoring server in the new runtime environment is to report to the hub in the model runtime environment, the IP port number must be the same for both runtime environments. |  
- RTE_TCP_HOST  
- RTE_TCP_STC  
- RTE_TCP_PORT |
| (Optional) Update the runtime environment data set high-level qualifiers.  
This actions is not required, because the new runtime environment name is appended to the high-level qualifier to make the data sets unique. Perform this action only if different high-level qualifiers are desired for this runtime environment. |  
- RTE_HILEV  
- RTE_VSAM_HILEV  
Scan the member for other occurrences of the existing high-level qualifiers to determine whether those variable values must be updated. |
| Change the monitoring server from hub to remote. | KDS_CMS_TYPE |
| Change the monitoring server name. | RTE_CMS_NAME |
| Update the monitoring server TCP/IP communications information:  
- Host name  
- TCP/IP started task name  
- Bind to a specific interface  
- IP port for communications  
- Update the IP port for the appropriate zzzz protocol |  
- KDS_CMS_TCP_HOST  
- KDS_CMS_TCP_STC  
- KDS_CMS_TCP_KDEBLST  
- KDS_CMS_TCP_zzzz_PORT |
| Have this remote monitoring server report to the SYSA hub monitoring server | KDS_HUB_ variables.  
These variables are commented out when the source batch parameter member has a hub monitoring server defined. Uncomment them to add information about the hub name and the protocols and ports used to connect to it. |
| Have all defined agents report to this remote monitoring server. |  
- Kpp_AGT_CMS_NAME (for agents that run in the monitoring server address space)  
- Kpp_CMS_NAM (for agents that run in separate address spaces)  
where pp is the 2-character product code. |
| Update the remote monitoring server TCP/IP communication information for agents that run in separate address spaces. | |
### Table 19. Actions required to update the runtime environment parameters (continued)

<table>
<thead>
<tr>
<th>Required actions</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Host name</td>
<td>• Kpp.CMS_TCP_HOST</td>
</tr>
<tr>
<td>• IP port</td>
<td>• Kpp.CMS_TCP_HOST</td>
</tr>
<tr>
<td></td>
<td>• where pp is the 2-character product code and zzzz is the defined IP protocol being used.</td>
</tr>
</tbody>
</table>

Update the agent TCP/IP communication information for agents that run in separate address spaces.

<table>
<thead>
<tr>
<th>Required actions</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Host name</td>
<td>• Kpp.CMS_AGT_HOST</td>
</tr>
<tr>
<td>• IP port</td>
<td>• Kpp.CMS_AGT_HOST</td>
</tr>
<tr>
<td></td>
<td>• where pp is the 2-character product code and zzzz is the defined IP protocols being used.</td>
</tr>
</tbody>
</table>

To see definitions and valid values for all the variables (also called batch parameters), generate a parameter map. See “Obtaining parameter reports” on page 96.

### Step 4. Run the CICATB job to validate the changes

When you specify the SCAN option for the SUBMIT parameter, CICATB validates that the required variables are defined and that the defined values meet the syntax requirements for the respective variables.

To validate the changes you have made to the parameter member:

1. Make the following changes to the CICATB JCL:
   - Modify the BATCHLIB parameter to specify the library where the SYSB member was generated.
   - Modify the BATCHMEM parameter to specify SYSB.
2. Submit the job.

If you submit CICATB from ISPF Edit mode, exit this mode for the INSTJOBS data set. The job does not run until the INSTJOBS data set is free.

A return code of 0 means the job executed successfully. It does not mean that the validation found no errors. The CICATB job output and the CB#Rssss member of the &shilev.INSTJOBS data set contain a report of the results. The heading of the report is similar to the following screen:

```
************************************************************************
* DATE: 2008/09/22                                               *
* TIME: 16:06                                                    *
* RTE NAME: RUN: 00/22/2008 16:06:39                            *
* INSTLIB: TEST.V621.INSTLIB                                      *
* BATCHLIB: TEST.V621.INSTJOBS                                    *
* BATCHMEM: SYSB                                                 *
* SUBMIT: SCAN                                                   *
*                                                          *
* THIS REPORT CONTAINS THE FOLLOWING SECTIONS:                   *
* 1. PARAMETER ERRORS                                            *
* 2. PARAMETERS YOU HAVE CHANGED                                 *
* 3. JOBS SORTED BY GENERATION SEQUENCE                         *
* 4. JOBS SORTED BY MEMBER NAME                                 *
* 5. COMPONENTS CONFIGURED IN THIS RTE                           *
************************************************************************
```

For the SCAN option, only the first two sections are created.

- **Parameter Errors**
This section contains errors the scan detected. Here is an example of errors that might be detected when a hub monitoring server is changed to a remote:

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>INPUT ERROR NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDS_CMS_HTTP_PORT</td>
<td>431</td>
<td>This parameter may only be enabled by a Hub TEMS.</td>
</tr>
<tr>
<td>KDS_CMS_CTSOAP</td>
<td>431</td>
<td>This parameter may only be enabled by a Hub TEMS.</td>
</tr>
<tr>
<td>KDS_HUB_GLB_APPL</td>
<td>431</td>
<td>Missing required parameter</td>
</tr>
</tbody>
</table>

The first two errors occur because these parameters do not apply and must be removed. The third error occurs because a parameter that is now required has not been defined.

**Parameters You Have Changed**

This section contains all parameters whose values have changed from the default values. It is a listing of all the parameters that have been specified, not just the parameters that were changed when the member was updated. Here is a portion of the section:

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>YOUR VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTE_DESC</td>
<td>420 and 621 RUNTIME ENV SYSB (none)</td>
</tr>
<tr>
<td>RTE_TYP</td>
<td>SHARING</td>
</tr>
<tr>
<td>RTE_SHARE</td>
<td>SMP</td>
</tr>
<tr>
<td>RTE_HILEV</td>
<td>PROD.V621</td>
</tr>
<tr>
<td>RTE_VSAM_HILEV</td>
<td>PROD.V621</td>
</tr>
<tr>
<td>RTE_VOL</td>
<td>OMEGXE</td>
</tr>
<tr>
<td>RTE_VSAM_VOL</td>
<td>OMEGXE</td>
</tr>
</tbody>
</table>

Review the report and correct any errors, and then run the CICATB SCAN again. When there are no errors in the Parameter Errors section, the parameter member can be used to create the new runtime environment successfully.

**Step 5. Create and (optionally) execute the batch jobs to create the runtime environment**

To create the batch jobs that create the runtime environment, but not submit them for execution, submit the CICATB job again with the SUBMIT(NO) option. In this scenario the systems share DASD, so you can execute CICATB to create the jobs from either the current system (SYSA) or the target system (SYSB).

Depending on the outcome, expect one of the following results:

If you did not correct all of the errors identified in the CICATB(SCAN) report, the jobs are not created and the job output contains the following message:

```
****************************************************
*** BATCH: SCANNER ERROR 10100 FROM KCICRTE
***
*** NOTE: JOB CREATION CANNOT PROCEED WHEN THERE
*** ARE ERRORS IN THE PARAMETER DECK.
```
You must make the Parm Deck error free before the RTE jobs can be created.

15:10:25 JOB: CB#R  DESC: PARAMETER SCAN REPORT

If there are no errors when the job is run, the job output contains a listing of the jobs created in the INSTJOBS data set and any warnings generated. The number varies based on the number of products and the product configuration options selected. For this scenario, 29 jobs were created, resulting in the following output:

** BATCH: PRODUCT LIST IS DF550 DS621 M2420 M5420 S3420 **

** 15:20:43 JOB: CB#1SBR  DESC: ALLOCATE RUNTIME LIBRARIES **

** 15:20:45 JOB: DS#LSBR  DESC: ASM, LINK VTAM LOGMODE TABLE **

** 15:20:46 JOB: DS#3SBR  DESC: CREATE RUNTIME MBRS **

** 15:20:48 JOB: PD#PSBR  DESC: CREATE PDS MBRS **

** 15:20:49 JOB: PD#QSBR  DESC: ALLOC AND INIT PDS **

** 15:20:52 JOB: DF#3SBR  DESC: CREATE RUNTIME MBRS **

** 15:20:54 JOB: DF#4SBR  DESC: REGISTER PRODUCT TO THE TEMS **

** 15:20:56 WRN: KPD61CPX GENHIST LIBS MISSING **

REASON: data set NOT FOUND. You have enabled the OMEGAMON II for SMS product to collect historical data at the TEMS.

The TEST.V621.SYSB.RGENHISx libraries are required to store the Data Warehousing data. These libraries are typically allocated and initialized during TEMS configuration. Default PDS control members (KPDPG and KPDAL) will be provided. Please refer to the DF#PSBR job for more information.

** 15:20:56 JOB: DF#PSBR  DESC: CREATE PDS MBRS **

** 15:20:57 JOB: DF#QSBR  DESC: ALLOC AND INIT PDS **

** 15:20:59 JOB: M2#5SBR  DESC: ALLOCATE ADDITIONAL data setS **

** 15:21:00 JOB: M2#3SBR  DESC: CREATE RUNTIME MBRS **

** 15:21:01 JOB: M2#4SBR  DESC: REGISTER PRODUCT TO THE TEMS **

** 15:21:03 JOB: M2#OSBR  DESC: MODIFY MENU SYSTEM SECURITY **

** 15:21:07 JOB: M5#4SBR  DESC: REGISTER PRODUCT TO THE TEMS **

** 15:21:09 JOB: M5#ISBR  DESC: CREATE RUNTIME MBRS, AGT TEMS **

** 15:21:14 WRN: KPD61CPX GENHIST LIBS MISSING **

REASON: data set NOT FOUND. You have enabled the OMEGAMON XE on z/OS product to collect historical data at the TEMS.

The PROD.V621.SYSB.RGENHISx libraries are required to store the Data Warehousing data. These libraries are typically allocated and initialized during TEMS configuration. Default PDS control members (KPDPG and KPDAL) will be provided. Please refer to the M5#PSBR job for more information.

** 15:21:14 JOB: M5#PSBR  DESC: CREATE PDS MBRS **
The first line in the listing identifies the products that are being configured in the runtime environment by their 2-character product code and version.release.modification level.

The output contains several warnings, all related to missing data sets required for historical data collection. These data sets are created, if they do not already exist, by the first job that runs and produces a warning.

The CB#PSBR and CB#NSBR jobs create sample JCL members in the &rhilev.&rte.RKANSAMU data set for copying the created started tasks procedures and VTAM nodes to PROCLIB and VTAMLST, respectively. No actual copying is performed. The sample jobs can be edited and used if desired. The SUB#ssss job (where ssss is the JCL suffix for the runtime environment) is a CLIST created to control the job submissions. The CLIST submits the jobs in the correct order. It also contains optional jobs that are commented out, as shown in this partial listing of the CLIST contents:

```plaintext
PROC 0 CANJOBS(TEST.V621.INSTJOBS)
CONTROL NOMSG NOFLUSH ASIS
/* ALLOCATE RUNTIME LIBRARIES
/* ** THIS JOB ALLOCATES EVERY
/* ** LIB REQUIRED IN THIS RTE
SUBMIT ('&CANJOBS(CB#1SBR)')
/* OPTIONAL: ASM, LINK VTAM LOGMODE TABLE
/* ** OPTIONAL JOB;
/* RUN IF REQUIRED
/*SUBMIT ('&CANJOBS(DS#LSBR)')
/* CREATE RUNTIME MBRS
```

Chapter 8. Configuration Tool (ICAT) deployment scenarios 159
Executing the CLIST creates the SYSB runtime environment and configures the products according to the contents of the SYSB batch parameter member. You can also submit the configuration jobs manually in the order specified in the CLIST.

To submit the jobs automatically after the CICATB job runs, instead of executing the CLIST or submitting them manually, specify SUBMIT(YES) or SUBMIT(RERUN) in the CICATB parameters. Both of these options create the same jobs as the SUBMIT(NO) option, and write each job to the INTRDR JES2 destination and to the &shilev.INSTJOBS data set.

- If you specify SUBMIT(YES), automatic submission occurs only if the jobs do not already exist in the &shilev.INSTJOBS data set.
- If you specify SUBMIT(RERUN), automatic submission occurs even if the jobs already exist in the &shilev.INSTJOBS data set. You can use this option to avoid deleting the existing configuration jobs before reconfiguring the runtime environment in batch mode.

Important: If the runtime environment contains any user-modified elements and if you intend to run the CICATB job with SUBMIT(RERUN) or SUBMIT(YES) specified, back up the runtime libraries before running CICATB.

When the jobs are submitted automatically, the CICATB output contains a record of their submission, as shown here:

```
READY
ISPSTART CMD(EX 'TEST.V621.INSTLIB(KINSTALL)'
'SYSTESTLIB(TEST.V621.INSTLIB) BATCHLIB(TEST.V621.INSTJOBS) BATCHMEM(SYB) SUBMIT(YES)')
IDC0549I MEMBER KCICTEMP DELETED
***
*** BATCH: PRODUCT LIST IS DF550 DS621 M2420 M5420 S3420
***
*** 11:41:12 JOB: CB#1SBR DESC: ALLOCATE RUNTIME LIBRARIES
***
*** 11:41:14 JOB: DS#LSBR DESC: ASM, LINK VTAM LOGMODE TABLE
***
*** 11:41:16 JOB: DS#3SBR DESC: CREATE RUNTIME MBRS
***
*** 11:41:19 JOB: PD#PSBR DESC: CREATE PDS MBRS
***
*** 11:41:20 JOB: PD#QSBR DESC: ALLOC AND INIT PDS
***
*** 11:41:25 JOB: DF#3SBR DESC: CREATE RUNTIME MBRS
***
*** 11:41:28 JOB: DF#4SBR DESC: REGISTER PRODUCT TO THE TEMS
***
*** 11:41:31 WRN: KPD61CPX GENHIST LIBS MISSING
*** REASON: data set NOT FOUND. You have enabled the
*** OMEGAMON II for SMS product to collect
*** historical data at the TEMS.
***
*** The TEST.V621.SYSB.RGENHISx
*** libraries are required to store the Data Warehousing
```
These libraries are typically allocated and initialized during TEMS configuration. Default PDS control members (KPDG and KPDAL) will be provided. Please refer to the DF#PSBR job for more information.

11:41:31 JOB: DF#PSBR DESC: CREATE PDS MBRS
11:41:34 JOB: DF#QSBR DESC: ALLOC AND INIT PDS
11:41:37 JOB: M2#5SBR DESC: ALLOCATE ADDITIONAL data setS
11:41:39 JOB: M2#3SBR DESC: CREATE RUNTIME MBRS
11:41:42 JOB: M2#4SBR DESC: REGISTER PRODUCT TO THE TEMS
11:41:44 JOB: M2#OSBR DESC: MODIFY MENU SYSTEM SECURITY
11:41:49 JOB: M5#4SBR DESC: REGISTER PRODUCT TO THE TEMS
11:41:53 JOB: M5#ISBR DESC: CREATE RUNTIME MBRS, AGT TEMS
11:41:59 JOB: M5#PSBR DESC: CREATE PDS MBRS
11:42:01 JOB: M5#QSBR DESC: ALLOC AND INIT PDS
11:42:05 JOB: S3#4SBR DESC: REGISTER PRODUCT TO THE TEMS
11:42:09 JOB: S3#ISBR DESC: CREATE RUNTIME MBRS, AGT TEMS
11:42:14 JOB: S3#PSBR DESC: CREATE PDS MBRS
11:42:16 JOB: CN#3SBR DESC: INSTALL CANDLE SUBSYSTEM
11:42:18 JOB: CB#8SBR DESC: CREATE SINGLE VTAM NODE
11:42:20 JOB: CB#PSBR DESC: SYSTEM PROCEDURE COPY JOB
11:42:23 JOB: CB#NSBR DESC: SYSTEM VTAMLST COPY JOB
11:42:26 JOB: CB#2SBR DESC: LOAD RUNTIME LIBRARIES
11:42:28 JOB: STA4SBR DESC: BUILD USER MODIFIED ELEMENTS
11:42:29 JOB: SUB#SBR DESC: CLIST TO DRIVE JOB SUBMISSION
11:42:29 JOB: CB#RSBR DESC: PARAMETER SCAN REPORT

The time stamps indicate when the job was submitted, not necessarily when it ran. The jobs are submitted with the same job name, so they execute in sequence. If any job scheduling is in place that might cause the jobs not to execute in sequence, use the SUBMIT(NO) option and submit the jobs manually. The CICATB job ends before the runtime environment creation jobs complete. Monitor the SYSLOG/OPERLOG to see the job processing and to identify any unexpected return codes (anything other than 0 or 4). This example shows a portion of the SYSLOG/OPERLOG messages, where jobname is the jobname used for the jobs submitted by CICATB.
For this submission, runtime environment creation jobs completed about 2 minutes after the CICATB job ended.

**Step 6. Customize and complete the configuration**

If all jobs have completed with 0 or 4 return codes, the runtime environment has been successfully created. You can now view and modify it from the Configuration Tool. Start the Configuration Tool and select a product to configure. The list of runtime environments now includes the newly created SYSB runtime environment [Figure 49].

![Figure 48. SYSLOG/OPERLOG messages](image)

Figure 48. SYSLOG/OPERLOG messages

For this submission, runtime environment creation jobs completed about 2 minutes after the CICATB job ended.

**Step 6. Customize and complete the configuration**

If all jobs have completed with 0 or 4 return codes, the runtime environment has been successfully created. You can now view and modify it from the Configuration Tool. Start the Configuration Tool and select a product to configure. The list of runtime environments now includes the newly created SYSB runtime environment [Figure 49].

![Figure 49. Runtime Environments (RTEs) panel with new SYSB runtime environment](image)

Figure 49. Runtime Environments (RTEs) panel with new SYSB runtime environment

You can now perform further customization. For example, if OMEGAMON XE on z/OS is installed, you might want to improve performance by moving the sysplex proxy from the hub on SYSA to the remote on SYSB. Additionally, you must perform any steps required to complete the configuration outside the Configuration Tool.
Now you can enable the SYSB runtime environment by performing the post-configuration procedures required for the products defined in the new runtime environment. Use the Configuration Tool to review the complete the configuration steps for each of the products in the environment, or to generate a file that consolidates the post-configuration instructions for all the products in the runtime environment. See also Chapter 6, “Completing the configuration,” on page 103.

To generate the consolidated instructions, complete the following steps:

1. From the Configuration Tool Main Menu (Figure 27 on page 79), enter 3 (Configure products).
2. On the Product Selection Menu, select any product.
3. On the Runtime Environments (RTEs) panel, enter Z (Utilities) next to the runtime environment in which you want to complete product configuration.
4. On the RTE Utility Menu (Figure 52 on page 175), enter 9 (Create Consolidated Complete the Configuration instructions).
   This option creates JCL containing a consolidated list of complete the configuration instructions for all products configured in the runtime environment. The list is stored in the &shilev.INSTJOBS data set, in a member named DFI@ssss (where ssss is the unique JCL suffix for the runtime environment).
5. Perform the tasks listed in the complete the configuration instructions. The list is generated dynamically and tailored specifically to your configuration.

   The most common tasks for configuring a monitoring server are covered in Chapter 6, “Completing the configuration,” on page 103. For detailed information about completing the configuration of monitoring agents, see the configuration documentation for each monitoring agent in the Tivoli Monitoring and OMEGAMON XE Information Center at http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?toc=

### Tips

- You do not have to configure the monitoring server in a runtime environment more than once, even though the monitoring server shows up on the Product Component Selection Menu for every monitoring agent.
- The Complete the configuration list for every monitoring agent on z/OS includes the task of registering the monitoring agent with the monitoring server in its runtime environment. See IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS.
Appendix A. Enabling system variable support

This appendix provides instructions for enabling system variable support and using it to run your installed products on any z/OS system. With system variables, the software becomes z/OS system-independent. The software can then be ported and started on any z/OS system without extensive reconfiguration.

System variables are elements that allow systems to share parameter definitions while retaining unique values in those definitions. System variables act like variables in a program; they can take on different values, based on the input to the program. When you specify a system variable in a shared parameter definition, the system variable acts as a placeholder. Each system that shares the definition replaces the system variable with a unique value during initialization.

If you use system variables, the components inherit the system values of the system on which they are started (the host z/OS system). These system-specific values are then automatically loaded into temporary data sets that exist only while the component runs. The result is that the software runs correctly by using the system-specific parameter values for the host z/OS system.

Tips

- You cannot use system variables in the runtime environment of a high-availability hub. See IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS for details.
- Product started tasks contain a preprocessing step that resolves all system variable specifications in the product parameter members.
- If you enable system variables, product parameter members contain many variables whose values are resolved during startup of the started task.

See the z/OS MVS Initialization and Tuning Reference (SA22-7592) for basic information on system variables.

Both the PARMGEN configuration method and the Configuration Tool method support system variables.

- For the PARMGEN configuration method, follow the instructions in Chapter 4, “Configuring products using the PARMGEN Workflow user interface,” on page 55.
- For the Configuration Tool (ICAT) method, follow the instructions in Chapter 5, “Configuring products with the Configuration Tool (ICAT),” on page 77.

Using the PARMGEN Workflow user interface to set up system variables

Complete these basic steps to enable and use system variable support within the runtime environment by using the PARMGEN Workflow user interface. For more information about using system variables in your PARMGEN configuration, see the scenarios described in the IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: PARMGEN Reference.

Prerequisites

Take care when specifying user-defined symbols.

The following PARMGEN profile parameters support symbolics as their parameter values. The parameter values are referenced in the STC PROC symbols:

**RTE NAME**

Used as the STC PROC SYS=symbol parameter.
RTE_HILEV
Used as the STC PROC RHILEV= symbol parameter.

RTE_VSAM_HILEV
Used as the STC PROC RVHILEV= symbol parameter.

RTE_X_HILEV_SHARING
Used as part of the STC PROC BASEHLEV= symbol parameter.

RTE_SHARE
Used as part of the STC PROC BASEHLEV= symbol parameter.

Any PARMGEN profile parameters that have VTAM APPLs (*_VTAM_APPL_*) are applicable to &stc_prefix.M2RC and &stc_prefix.TOM, as well as the STC PROC APPL= symbol parameter.

Additional symbolics for OMEGAMON XE for IMS on z/OS sites:

KI2_I1nn_CLASSIC_IMSID
Used in the OMEGAMON for IMS (3270) STC PROC IMSID= symbol parameter.

KI2_I1nn_CLASSIC_MPFREUX
Used in the OMEGAMON for IMS (3270) STC PROC MPFREUX= symbol parameter.

Note: Exercise caution when specifying user-defined symbols (as opposed to static symbols) for any of these CONFIG profile parameters because the JCL rules that govern the use of system symbols state that most system symbols cannot be used in batch JCL. In started-task procedures, you will generate a startup JCL error if you issue the /START&stc command, as user symbols (&SYSCLONE., for example) referenced as part of the STC PROC statement are not recognized by the system (unlike &SYSNAME.or &SYSCLONE: because these are system-wide defined static symbols).

Use one of the following ways to work around the JCL rule:

1. By overriding the value of these symbols by using the startup command
2. By using the PARMLIB sample composite STC startup member (CANSAPF by default) and then defining your own PROC symbol overrides

For VTAM_APPL-related parameters, no exceptions are allowed to the JCL rule. You must not use user-defined symbols or both the started tasks and the VTAM major node members will not work.

Enable system variable support
In [Step 5. Customize PARMGEN configuration profiles on page 64], select option 1 to produce the list of parameters for this runtime environment. To enable system variable support, you must change the value of the RTE_SYSV_SYSVAR_FLAG parameter to Y as part of your customization of the parameter list.

Optionally, if you are configuring a runtime environment that will use and resolve LPAR system variables, also edit the value of the RTE_X_SYSV_OVERRIDE_SYMBOLS parameter to Y. Editing this parameter causes LPAR-specific values specified in parameters (such as system names) to be resolved correctly in the runtime members.

You can then customize the other parameters within the member to use the system variables and user-defined variables. For example, you can modify the runtime environment name by using a system variable:

RTE_NAME OMXE$SYSNAME.

You can also customize parameters to employ user-defined symbols and variables. In the following example, the USS directory used can be represented by a variable:

RTE_USS_RTEDIR "&RTE_USS_DIR."

On completion of the parameter customization, save the changes and return to CUSTOMIZE PARMGEN CONFIGURATION PROFILE MEMBERS panel.
Specify user-defined symbols and variables

On the Customize PARMGEN Configuration Profile Members (KCIP@PG6) panel, select option 3 to customize the System Variables profile to define all the user-defined symbols added in the previous step. Figure 50 shows an example of the member where the resolved symbols are defined.

Figure 50. Pre-defined and user-defined symbolics

On completion of the parameter customization, save the changes and return to CUSTOMIZE PARMGEN CONFIGURATION PROFILE MEMBERS panel. You can continue configuring the runtime environment as described in Chapter 4, “Configuring products using the PARMGEN Workflow user interface,” on page 55.

Using the Configuration Tool (ICAT) to set up system variables

These are the basic steps of enabling and using system variable support with the Configuration Tool:

1. Define and configure a runtime environment. During runtime environment configuration, specify the values to enable system variable support (Use z/OS system variables? on the second Add or Update Runtime Environment panel: see “Enable system variable support” on page 168).

Note: Multiple runtime environments might be required depending on the runtime environment structure (full, sharing, base, or SMP/E), the type of monitoring server (hub or remote), and variations in product mixtures.

2. Create the system variable parameter member.
After configuring a runtime environment, you can create the system variable parameter member. Each runtime environment contains one user system variable parameter member named RKANPARU, which contains all system-specific values.

See “Create the system variable parameter member” on page 169 for information on creating the system variable parameter member in the &rhilev.&rte.RKANPARU library.

3. Create the VTAM major node rename job.

After configuring a runtime environment, you can create the VTAM major node rename job. This job creates VTAM major nodes on remote systems with names that are resolved from your system variable specification.

See “Create the VTAM major node rename job” on page 170 for information on creating a VTAM major node rename job.

4. Create the runtime environment transport job by using the RTE Utility option Generate Sample Transport JCL.

5. Copy the runtime environment to a remote system by using the runtime environment transport job (or any other copy utility).

6. After the copy completes, edit the system variable parameter member RKANPARU. If necessary, set values for components running on other systems. For example, set values for a hub monitoring server running on a different LPAR.

7. Perform other remote system setup tasks as required.
   - Copy the new started tasks to your system procedure library. These started tasks have been enabled for system variables.
   - If you are using VTAM system variable support, you must run the VTAM major node rename job. This job creates new major nodes that are named according to your system variable specifications. After the new nodes are created, copy them to SYS1.VTAMLST.
   - VSAM file allocation and seeding are required on every system.

8. Start the components.

Enable system variable support

You enable system variable support from the Add Runtime Environment panel. In an existing runtime environment, use the Update (U) command to enable system variable support.

Complete the following steps to enable system variable support.
1. From the Configuration Tool Main Menu (shown in Figure 27 on page 79), enter 3 (Configure products).
2. On the Configure Products panel, enter 1 (Select product to configure).
3. Select the product that you want to configure. The Runtime Environments (RTEs) panel is displayed.
4. If you are adding a new runtime environment, type A in the Action field and specify all other required information and press Enter. If you are updating a runtime environment, type U in the Action field and press Enter. The Add Runtime Environment or the Update Runtime Environment panel is displayed.
5. Specify any required values or accept the existing ones and press Enter.
   A second Add or Update Runtime Environment panel is displayed.
6. Specify the following values to enable system variable support:
   **Use z/OS system variables?**
   - Specify Y to enable support. The default is N
   **RTE name specification**
   - Specify the system variable name by which the runtime environment is identified in the
SYS1.PARMLIB LPAR system definition member. The default is &SYSNAME. This value becomes the value of the SYS parameter in all started tasks (for example, SYS='&SYSNAME').

**Note:** Resolved system variable values cannot exceed the length of the variable name (maximum length of 8 characters).

---

**Important**

If you change the status of system variable support in an existing runtime environment (on to off or vice versa), you must reconfigure all products in that runtime environment. This includes specifying VTAM values and creating runtime members.

---

7. (Optional) Specify the following values on the second Add Runtime Environment panel to enable other functions:

**RTE base alias specification**

If this runtime environment is sharing with a base runtime environment, specify an optional system variable specification for the base runtime environment. This value is inserted into the base runtime environment library references in all started tasks. The resolved name must be a valid library name qualifier. This field is commonly used to switch between base runtime environments at different maintenance levels. Use the runtime environment base alias to obtain the following benefits:

- An easy way to switch runtime environment bases.
- An alternative way to refer to an existing base.

**Note:** A label of n/a might be displayed next to this field if the current runtime environment is not sharing with a base runtime environment.

**Applid prefix specification**

Specify the VTAM applid prefix that contains system variables. Be sure to place a period after the last symbol in the specification. The resolved prefix can be a maximum of four characters. The default is K&SYSCLONE.

**Use VTAM model applids?**

If you want to use model applids (wildcards), specify Y. Model applids generate VTAM nodes that contain applids with wildcard suffixes wherever possible. Then you can use any applids that match the pattern within the VTAM node. The default is N.

8. When you have finished specifying the values to enable system variable support, press F3 until you return to the Main Menu.

---

**Create the system variable parameter member**

You enable system variable support from the Add Runtime Environment panel. In an existing runtime environment, use the Update (U) command to enable system variable support.

After configuring a runtime environment, you can create the system variable parameter member.

Each runtime environment contains one user system variable parameter member named RKANPARU(rte). All system-specific values are contained in this member.

Complete the following steps to create the system variable parameter member, rte, in the &rhilev.&rte.RKANPARU library:

1. From the Configuration Tool Main Menu (shown in Figure 27 on page 79), enter 3 (Configure products).
2. On the Configure Products panel, enter 1 (Select product to configure).
3. Select the product that you want to configure.
The Runtime Environments (RTEs) panel is displayed.

4. Type Z (Utilities) next to the name of the runtime environment you have configured and press Enter.

5. On the RTE Utility Menu, select Create System Variable Parameter Member and press Enter.
   The JCL that creates the system variable parameter member (CB#Vssss) job is displayed.

   **Important:** The KCIDSV41 step in the CB#Vssss job requires exclusive enqueue on the runtime libraries allocated in each DDNAME. Make sure that the job has the access it requires.

6. Review the JCL and submit the job. Verify that the job completes successfully and that all return codes are zero.

7. Press F3 to return to the Runtime Environments (RTEs) panel.

8. Edit the &rhilev.&rte.RKANPARU(rte) parameter member. Follow the directions to ensure correct resolution of cross-system variables.

Create the VTAM major node rename job

After configuring a runtime environment, you can create the VTAM major node rename job. This job creates VTAM major nodes on remote systems with names that are resolved from your system variable specification.

Complete the following steps to create a VTAM major node rename job in the INSTJOBS library.

1. From the Configuration Tool Main Menu (shown in Figure 27 on page 79), enter 3 (Configure products).
2. On the Configure Products panel, enter 1 (Select product to configure).
3. Select the product that you want to configure.
   The Runtime Environments (RTEs) panel is displayed.
4. Type Z (Utilities) next to the name of the runtime environment you have configured and press Enter.
5. On the RTE Utility Menu, select Create System Variable VTAM Major Node Rename Job, and then press Enter.
   The JCL that renames the VTAM major node (CB#7ssss) job is displayed.
6. Review the JCL. Do not submit the job yet.

   **Note:** You must submit this job on each remote system where the monitoring software is to run. The job is in the &rhilev.&rte.RKANSAMU library.

7. Press F3 to return to the Runtime Environments (RTEs) panel.

Create one VTAM major node for all products in the runtime environment

A single VTAM major node can contain all the VTAM applids for all of the TMS:Engine-based products you have configured in the runtime environment. This single major node is then used in place of the individual product major nodes.

If you choose to use a single VTAM major node, you must create it after all products have been configured in the runtime environment. After the node is created and copied to your system VTAM system library (SYS1.VTAMLST), you vary it active and then start all of the components (started tasks).

Complete the following steps to create a single VTAM major node in the &rhilev.&rte.RKANSAMU library.

1. From the Configuration Tool Main Menu (shown in Figure 27 on page 79), enter 3 (Configure products).
2. On the Configure Products panel, enter 1 (Select product to configure).
3. Select the product that you want to configure.
   The Runtime Environments (RTEs) panel is displayed.
4. Type Z (Utilities) next to the name of the runtime environment you have configured and press Enter.
5. On the RTE Utility Menu, select Create VTAM Major Node (one node for all products) and press Enter.
6. On the Create VTAM Major Node panel, type the name you want to use for the single node and press Enter.
   The JCL that creates the single node is displayed.
7. Review the JCL and submit the job. Verify that the job completed successfully and that all return codes are zero.
8. Press F3 to return to the Runtime Environments (RTEs) panel.
Appendix B. Configuration Tool reference

This appendix contains information about the services and utilities provided by the Configuration Tool. It also contains information about the Runtime Environments (RTE) panel and the actions and utilities associated with it:

- "Configuration services and utilities"
- "Runtime environment utilities" on page 174
- "Batch mode utilities" on page 178
- "The Runtime Environments (RTE) panel" on page 180

The following ISPF-related restrictions apply to the Configuration Tool:

- If you are using a 3270 Model 2 (24 x 80) display, you must turn off the predefined function (PF) keys so that the Configuration Tool panels are not truncated. To turn off the predefined function keys, type PFSHOW on any ISPF command-line and press Enter until the function keys are no longer displayed.
- You cannot use the ISPF feature for edit recovery. If you enter the ISPF RECOVERY ON command, edits produce a recovery error message. Enter the RECOVERY OFF command to suppress the error messages.

Configuration services and utilities

The configuration services and utilities are used to perform various services on a runtime environment and to collect diagnostic information. Some of the services modify the Configuration Tool values stored in ISPF tables. Do not modify any values unless you are told to do so in the documentation or by Software Support personnel. If the Configuration Tool values are modified incorrectly, the Configuration Tool might stop functioning or produce unpredictable results.

To access the configuration services and utilities:

1. On the Configuration Tool Main Menu panel enter 3 (Configure products).
   The Configure Products panel is displayed (Figure 31 on page 84).
2. Enter S (Services and utilities).
   The Configuration Services and Utilities panel is displayed, as shown in Figure 51.

![Figure 51. Configuration Services and Utilities panel](image)

Select any of the following options:

**Unlock high-level qualifier values**
You specify high-level qualifier values when you set up your configuration environment. If you have to modify these values later, you must first unlock them.
Attention: If you unlock and change the high-level qualifiers, the Configuration Tool does not automatically delete and reallocate the existing libraries. The jobs generated by the Configuration Tool fail if they are pointing to incorrect libraries.

DEBUG options
When IBM Software Support personnel request additional information to investigate a problem you have encountered with the Configuration Tool, use this option to specify or modify DEBUG parameter values.

Display an ISPF table
Use this option to view the contents of an ISPF table located in the data library. You can limit the information displayed for an ISPF table by specifying one to three sets of display criteria under Optional selection parameters. You can use these actions on this table:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>END (F3)</td>
<td>Go to previous record</td>
</tr>
<tr>
<td>ENTER</td>
<td>Go to next record</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Go back to prior panel. To cancel an action, you must type the word cancel. The usual End key (F3) does not function as usual here.</td>
</tr>
<tr>
<td>UP/DOWN</td>
<td>Use scroll variable</td>
</tr>
</tbody>
</table>

Execute a CLIST in the TKANCUS library
When IBM Software Support personnel request additional information to investigate a problem you have encountered with the Configuration Tool, they might ask you to execute a specific CLIST/REXX executable that is in the TKANCUS library. Specify the values requested by IBM Software Support personnel.

Prepare user libraries
The Prepare User Libraries utility supports the allocation of any user libraries added since the prior installed release. This utility displays the Runtime Environments (RTES) for Conversion panel. This panel lists all the runtime environments in this installation environment except base environments. On this panel, you can prepare user libraries for the default listing (all runtime environments) or delete the runtime environments you want to exclude from the conversion process.

The Configuration Tool generates a &shilev.INSTJOBS(KCIJSP01) job to allocate the applicable new user libraries for each runtime environment selected.

Runtime environment utilities
The runtime environment utilities perform various useful processes for maintaining a runtime environment and creating new runtime environments using batch mode.

Take the following steps to access the utilities:
1. On the Main Menu panel, select option 3 (Configure products). The Configure Products panel is displayed.
2. Select option 1 (Select product to configure). The Product Selection Menu panel is displayed.
3. Select a product that is configured in the runtime environment for which you want to access the utilities. The Runtime Environments (RTEs) panel is displayed.
4. Enter Z beside the appropriate runtime environment. The RTE Utility Menu is displayed on page 175.
Select any of the following utilities:

- **Create/Clone batch mode parameters**
  This utility creates a member that contains the configuration values for all the products in the selected environment. This member can be used to back up the configuration parameters of an existing runtime environment, restore a runtime environment based on a saved configuration, and reconfigure an existing runtime environment when something in the environment has changed.

  You can also clone the parameter member for an existing environment and modify it as necessary to create a new runtime environment by using the batch mode job, CICATB. You can make an exact duplicate of the current environment (Clone batch mode parameter deck, unmodified) or eliminate some of the products configured in the environment (Clone batch mode parameter deck, with modifications). You can have the Configuration Tool open the member after it is generated so you can edit the parameter values for the target environment.

  See Chapter 7, “Using the Configuration Tool (ICAT) batch mode to replicate a configured environment,” on page 121 for more information about the use of batch mode parameters. See “Step 2. Create and clone a batch mode parameter member” on page 125 for step-by-step instructions for using this utility.

- **Create the system variable parameters**
  This utility creates a system variable support parameter member for the selected runtime environment. Supported products resolve these parameters at execution time. z/OS system variable usage must be enabled for the selected runtime environment to execute this utility. See Appendix A, “Enabling system variable support,” on page 165 for instructions for using this utility.

- **Create system variable VTAM major node rename job**
  This utility creates a job named CB#VSHRA in the &rhilev.&rte.RKANSAMU library. The job creates new major nodes with names that are resolved from your system variable specification. The job must be submitted on every remote system where system variable support is being used. After the new nodes are created, they must be copied to SYS1.VTAMLST.

  See “Create the VTAM major node rename job” on page 170 for instructions on using this utility.

- **Create VTAM major node (one node for all products)**
  This utility creates a single VTAM major node in the &rhilev.&rte.RKANSAMU library that contains the VTAM applids for all of the IBM products you have configured in the runtime environment. This single major node is then used in place of the individual product major nodes.
Generate sample transport JCL
This utility generates sample batch mode configuration scenarios and transport runtime environment jobs and copies them to the &rhilev.&rte.RKANSAMU library.

See Chapter 7, “Using the Configuration Tool (ICAT) batch mode to replicate a configured environment,” on page 121 for descriptions of the sample transport jobs and scenarios and step-by-step instructions for generating and using the jobs.

Generate sample system procedure copy JCL
This utility creates sample JCL to copy system procedures from RKANSAMU to a user-specified system library for this runtime environment. You can specify any system library if you do not want to update your PROCLIB directly.

The Configuration Tool generates the sample IEBCOPY job named CB#Pssss, which creates the KCISYSPJB sample in the &rhilev.&rte.RKANSAMU library. The CB#Pssss job generates a compression step for the respective libraries if this runtime environment is not using the PDSE format. Before you submit the KCISYPJB job, ensure that you have authority to update the respective system libraries.

Generate sample system VTAMLST copy JCL
This utility creates sample JCL to copy VTAM major node members from RKANSAMU to a user-specified system library for this runtime environment. You can specify any system library if you do not want to update your VTAMLST directly. The default system library is SYS1.VTAMLST.

The Configuration Tool generates the sample IEBCOPY job named CB#Nssss, which creates the KCISYSNJB sample in the &rhilev.&rte.RKANSAMU library. The CB#Nssss job generates a compression step for the respective libraries if this runtime environment is not using the PDSE format. Before you submit the KCISYNJB job, ensure that you have authority to update the respective system libraries.

If you have enabled system variable support for this runtime environment, this option does not apply.

Analyze user modified elements
This process facilitates the identification of user modified elements and shows which of those elements are modified by a specific batch job. This process is not applicable to a remote runtime environment.

See Step 3. Adjust and validate the parameters” on page 128 for information on using analysis of user-modified elements before submitting batch mode jobs.

Create consolidated Complete the Configuration and README reports
This option concatenates the contents of the Complete the Configuration instructions for all products configured in a runtime environment into a single, printable report. The report is stored in the &shilev.INSTJOBS data set, in a member named DFI@ssss (where ssss is the unique JCL suffix for the runtime environment).

This option also consolidates all the README files in the &thilev.TKANCUS data set and stores them in a single file named $README in the &shilev.INSTJOBS data set.

Process Edit subcommands for INSTJOBS or RKANPARU members
This option applies a list of EDIT subcommands to configuration batch jobs in the &shilev.INSTJOBS data set, sample parameter decks in the &thilev.TKANCUS data set, or members of the &rhilev.&rte.RKANPARU data set.

Verify configuration and generate parameter map
This utility has two options:

• By using the Verify configuration option, you can verify the creation and successful execution of all configuration batch jobs, and the creation of all required runtime data sets and members. You can run the configuration verification job by selecting this option at any time; it is a good idea to run it as soon as you finish configuring a runtime environment and before you load the runtime libraries. Additionally, the configuration verification job is run automatically when you
load the runtime libraries. This job can be run in batch mode only; it cannot be run in foreground. If a large number of components and products are configured in the runtime environment, the configuration verification batch job might take a long time to run.

The configuration verification job is specific to the runtime environment. You must generate a unique configuration verification job for each runtime environment you want to verify. If the configuration of the runtime environment changes (for example, if new components are configured or existing components are reconfigured), be sure to regenerate and rerun the configuration verification job.

The configuration verification job generates a report that lists the required data sets and members; indicates the batch job that creates each one; identifies which data sets and members are missing, and which batch jobs are missing or did not complete successfully; and specifies which of the missing or unsuccessful batch jobs and which of the missing data sets and members are required and which are optional. This report is stored in the &shilev.INSTJOBS.IVP$s$ member, where $sss$ is the JCL suffix for the runtime environment. The job information in the report is retrieved from a configuration batch job log stored in the &shilev.INSTLOG data set. This log is cumulative for all jobs run to configure all runtime environments in the installation environment.

• The Generate parameter map option generates a parameter map for the runtime environment. If the map is generated before any products are configured in the runtime environment, a basic report is generated for all the components installed into the target libraries that were installed by SMP/E. The report gives a short description and other basic information about each parameter. If the map is generated after products are configured, the report also contains the value for each configured parameter and other post-configuration parameter information such as the name of the configuration job that stored each parameter and data about completion of each job and each step. The job and step completion codes reported in the parameter map are retrieved from the configuration batch job log stored in the &shilev.INSTLOG data set.

The parameter map is generated in foreground and is displayed as soon as it is complete. If a large number of components and products are configured in the runtime environment, generating the parameter map might take several minutes. The parameter map for a runtime environment is stored in the &shilev.INSTJOBS.IVP$s$ member.

A detailed parameter map requires data that is produced by the configuration verification job. Consequently, to produce a detailed parameter map for a runtime environment, you must first run the configuration verification job for the same runtime environment. If you subsequently reconfigure components or configure additional component in the runtime environment, regenerate and rerun the configuration verification job and then regenerate the detailed parameter map for the runtime environment.

Enter README IVP on the command-line for more information about the configuration verification report and the parameter map. You can also find more information about the parameter map in "Set appropriate parameter values" on page 96.

Upgrade attribute and catalog members

This option updates the catalog and attribute files in all runtime environments accessible to the Configuration Tool, by copying the most recently installed files from the &thilev.TKANDATV target data set that was installed by SMP/E to the runtime library &rhilev.&rte.RKANDATV for each runtime environment. For more information on catalog and attribute files, see the IBM Tivoli Monitoring: Installation and Setup Guide.

Generate health check configuration members

Use this option to configure the two health checks implemented by the Configuration Tool. One health check verifies, at a user-defined interval, that all started tasks configured for a runtime environment are running. The default interval is 30 minutes, and the valid range is 10 minutes to 1440 minutes (24 hours). The second health check, which is run at IPL, verifies the APF authorization of all load libraries that must be APF-authorized for a runtime environment. Each health check runs as a system REXX EXEC and enables a system alert if an exception is found.
Batch mode utilities

Three Configuration Tool utilities are available to assist you if you are using batch mode processing.

**KCISETUP**
Sets up the environment required for using the ISPF macros provided with the Configuration Tool.

**KCICFKEY**
Manages the PF keys that are used for the Configuration Tool batch utilities.

**KCICPGHP**
Displays help information for parameters in a batch parameter deck member.

These utilities are designed to run outside the Configuration Tool, but can also be used while in the Configuration Tool.

**KCISETUP: Set up the environment for batch utilities**

Use the KCISETUP utility to set up the environment that is necessary for using the other Configuration Tool batch utilities. This utility must be run once after starting your TSO ISPF session and can be run only from an ISPF session.

Before using the KCISETUP utility, you must generate the KCISETUP member in your `$shilev.INSTLIB` data set. KCISETUP can be run only once per session. No confirmation message is issued to indicate successful completion of KCISETUP.

Follow these steps to generate KCISETUP.

1. Start the Configuration Tool.
2. On the Main Menu panel select option 3 (Configure products).
   The Configure Products panel opens.
3. Select option 5 (Services and utilities).
   The Configuration Services and Utilities panel opens.
4. Select option 2 (Create batch mode job).
   Wait until you see the message “CICATB JOB CREATED” in the upper-right corner of the panel.

   **Tip**
   The default location for the CICATB job is the INSTJOBS library. To see the complete path of the library the batch mode job was created in, press F1. This displays a text box with the complete path at the bottom of the panel. The text box is shown only if F1 is pressed immediately after the option 2 (Create batch mode job) was selected. Otherwise, the general help for the panel is shown.

5. Press F3 until you return to the main menu.

**Note:** KCISETUP must be created once on an image and can be used for all subsequent parameter deck processing on that image. If the ISPF environment on the new image is different the one on the model image, you must recreate KCISETUP.

Invoke the environment setup utility using either of the following methods:

- From the ISPF command-line enter TSO EXEC `$shilev.INSTLIB(KCISETUP)`
- From the ISPF Primary Option Menu, select Enter TSO or Workstation commands and enter
  `EXEC `$shilev.INSTLIB(KCISETUP)``
KCICFKEY: Manage ISPF PF key

Use the KCICFKEY utility to manage ISPF session PF keys that are used for batch utilities. This includes turning the PF keys on and off, and toggling which set of keys display. This utility can be run only under an ISPF session.

If you plan to use KCICFKEY to manage the ISPF session PF keys for the batch utilities, you must turn on the predefined function keys. To do this, issue the PFSHOW command from either the ISPF command-line or any of the Configuration Tool panel command-lines.

Before using this Configuration Tool batch utility, you must use the KCISETUP utility to set up the environment.

To use the KCICFKEY utility, your ISPF session must support 24 PF keys. Follow these steps to set up your ISPF session to support 24 PF keys.

1. From the ISPF Primary Option Menu under Settings, select Terminal and user parameters > Function keys > Non-Keylist PF Key settings.
2. In the Number of PF Keys field, type 24 and press F3 to return to the ISPF Primary Option Menu.

If the ISPF session is not set up to support 24 PF keys, the KCICFKEY utility issues the following ISPF warning message:

PFKEYS COUNT ERROR. Number of PF Keys must be 24. See ISPF Settings.

The Configuration Tool PF Key Manager owns PF keys 13 through 24. On keyboards that do not support 24 PF keys, PF keys 13-24 are obtained by holding the Shift key and pressing a function key. While the Shift key is pressed, function keys 1-12 become 13-24. After you exit the Configuration Tool PF Key Manager, all of your prior PF key and Show State settings are restored.

Invoke the PF Key Manager using one of the following methods:

- From the ISPF Edit command-line, enter
  KCICFKEY state
  To use this method you must be running under an ISPF session, editing a member or data set.
- From the ISPF command-line, enter
  TSO KCICFKEY state

where state is the desired state of the Configuration Tool PF keys. The valid states are:

ON       Turns on the PF keys.
OFF      Turns off the PF keys.
SHOW    Toggles the PF keys (if active) among All (1–24), Alternate (13–24), and Primary (1–12).
HELP     Displays the PF Key Manager help information.

If state is not specified, the Configuration Tool PF keys toggle between ON and OFF.

The PF Key Manager sets the following keys for the batch utilities:

F13       Displays the PF Key Manager Help information.
F14       Provides detailed help information for the batch parameter on the line where the cursor is positioned in the batch parameter deck.
F15       Turns off the PF keys.
F21       Toggles the PF keys shown at the bottom of the display among All (1-24), Alternate (13-24), and Primary (1-12).
**KCICPGHP: Display help for configuration parameters**

Use the KCICPGHP utility to display help information for parameters in a batch parameter deck member. The detailed help information is used to help you modify or construct a batch parameter deck. This utility must be run from an ISPF Edit session.

Before using this batch utility, you must use the KCISETUP utility to set up the environment.

Invoke the help utility using either of the following methods:

- From an ISPF Edit command-line when editing the batch parameter deck, enter the command KCICPGHP, position the cursor on the row that contains a batch parameter, and then press Enter.
- Position your cursor on the row that contains a batch parameter and then select the PF key assigned by the Configuration Tool PF Key Manager.

With either method, you must position the cursor on the row that contains the batch parameter. The utility then isolates the parameter, performs a lookup, and displays a pop-up window with the detailed help information.

The batch parameter help contains the sections shown in Table 20.

### Table 20. Batch parameter format

<table>
<thead>
<tr>
<th>Parameter section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Information</td>
<td>The parameter name, followed by a short description of the parameter.</td>
</tr>
<tr>
<td>Description Area</td>
<td>The detailed help information for the parameter. This area is scrollable, as indicated by the (+) indicator on the bottom right. F7 and F8 are assigned to scroll this area.</td>
</tr>
<tr>
<td>Attribute Area</td>
<td>The attributes of the parameter. This information can help you determine what type of data is expected for this parameter.</td>
</tr>
<tr>
<td>Key Area</td>
<td>The function key assignments that apply only to the pop-up window. <strong>Note:</strong> F5 (Show All) displays the help information for all parameters that make up this product.</td>
</tr>
</tbody>
</table>

**The Runtime Environments (RTE) panel**

The Runtime Environment panel in the Configuration Tool, shown in Figure 53, is one of the primary recurring panels in all configuration operations.

**Figure 53. Runtime environments panel**

Select the appropriate action:
A (Add RTE)
Defines the runtime environment to the Configuration Tool. This option creates a definition for the Configuration Tool only. For step-by-step instructions on adding a runtime environment, see 2 on page 85.

B (Build libraries)
Generates a job to allocate the required runtime libraries for the selected product. This job is presented for your review and submission. Action B must be performed for every runtime environment that will contain the selected product. This includes base runtime environments. For step-by-step instructions, see 3 on page 95.

C (Configure)
Presents panels to collect the parameter values required to configure the selected product for this runtime environment. Default values are used when applicable. Action C must be performed after Action B for every runtime environment that contains the selected product. You do not have to perform action C for a base runtime environment.

L (Load all product libraries after SMP/E)
Loads all product libraries after SMP/E maintenance or product configuration. This action generates a job to load the members of the target libraries that were installed by SMP/E into the runtime environment libraries.

D (Delete)
Deletes the runtime environment. The runtime environment definition and associated configuration values are removed from the Configuration Tool and a batch job is created to delete the libraries. See “Deleting a runtime environment” on page 183 for more information and instructions.

U (Update)
Displays current runtime environment values and indicates those you can change. To apply changes, you must use actions B, C and L in sequence for the affected products. You must also complete any configuration steps required outside the Configuration Tool, such as copying renamed procedures to your PROCLIB.

V (View Values)
Displays current runtime environment values.

Z (Utilities)
Displays a list of installed utilities used to perform various useful processes for maintaining the runtime environment. (See “Runtime environment utilities” on page 174 for descriptions.)

R (README Table of Contents)
Displays a list of available README files for selection. (Figure 54 on page 182.)

Select the README file of interest. If you select AAA, the Configuration Tool displays information about new features that affect the configuration of all components and products installed in the CSI.

To display a README file from any other Configuration Tool panel, enter README ccc (where ccc is the 3-character code for the type of README file) at the command-line.
Table 21 shows the actions that must be performed when new products or maintenance for existing products are installed.

<table>
<thead>
<tr>
<th>Installation and maintenance tasks</th>
<th>Add RTE (A)</th>
<th>Build libraries (B)</th>
<th>Configure (C)</th>
<th>Load all libraries after SMP/E (L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>You installed new products or new versions of existing products and you want to create a new runtime environment.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>You installed new products and you want to use an existing runtime environment.</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>You installed maintenance for existing products, and the PSP bucket file indicates that the configuration settings for the product must be changed.</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>You installed maintenance for existing products, and the PSP bucket file indicates that no changes to the configuration settings are required.</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Note:** For sharing-with-SMP/E runtime environments, you are not required to reload the libraries unless the action is specifically indicated by ++HOLD information on a Program Temporary Fix (PTF).

If you are installing a new version of an existing product in an existing runtime environment, you do not have to perform the Build and Configure steps unless you want to take advantage of new features that are not enabled by default. You only have to load the libraries. If you are upgrading a Tivoli Enterprise Monitoring Server, you must also run the monitoring server migration job, DS#Mssss. For more information, see the *IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Upgrade Guide*.

**Loading runtime environments**

The **Load All Product Libraries After SMP/E** option generates JCL to load the runtime libraries for this runtime environment from the target libraries that were installed by SMP/E.

The runtime environment libraries must be loaded under following conditions:

- after initial product configuration
- after SMP/E maintenance
- after configuring a new product into an existing runtime environment
In a sharing configuration, only the data sets for the master (shared) full or base runtime environment must be loaded after SMP/E maintenance is applied, because these types of runtime environments contain copies of the target libraries that were installed by SMP/E. In a sharing-with-base or sharing-with-full configuration, the data sets for the master runtime environment are loaded automatically when the Load option is selected for the sharing runtime environment.

If the runtime environment is set up to share the target libraries that were installed by SMP/E, no load of the runtime environment libraries is required after maintenance.

**Deleting a runtime environment**

If you no longer require a particular runtime environment, including the runtime libraries and the configuration values set for the runtime environment, you can delete it.

Confirm that this runtime environment is no longer required before proceeding. If you delete a runtime environment, any other runtime environment that shares libraries with it becomes inoperable. The delete job deletes all libraries that match one of the following patterns:

- `&rhilev.&rte.*`
- `&rvhilev.&rte.*`

Manually delete any libraries that you allocated for this runtime environment with a different high-level prefix. Also confirm that any libraries that match the pattern `&rhilev.&rte.*` or `&rvhilev.&rte.*` for a base runtime environment (which might not be part of the runtime environment) are renamed if you do not want to delete those libraries.

As a precaution, back up the entire SMP/E and runtime environment, including the `&shilev.INST*` libraries. This allows the runtime environment and the installation libraries to be restored as required. Restoring the runtime environment alone is not sufficient because the runtime environment information is deleted from these installation libraries.

Follow these steps to delete a runtime environment.

1. Enter D next to the runtime environment you want to delete and press Enter. This displays the Delete the Runtime Environment panel shown in [Figure 55](#).

---

**Figure 55. Delete runtime environment panel**

On the Delete Runtime Environment panel, enter Y in response to the following question:

Are you sure you want to delete this RTE?
2. Review the JCL and submit the job. Check the line `Delete &rhilev.&rte.*`. Ensure that this applies only to your libraries. This job deletes all libraries that match the pattern.

3. Press F3 to return to the Main Menu.

**Important**

After you respond Y on Figure 55 on page 183 and the JCL for deleting the runtime environment is generated, you are committed to deleting the runtime environment. If you back out of the JCL without submitting it, the runtime environment is removed from the Runtime Environments (RTEs) panel, but the data sets for that runtime environment are not deleted and these data sets are no longer accessible from the Configuration Tool. If you did not submit the JCL when it was presented, locate the delete job in the `&shilev.INSTJOBS` data set, and submit the job manually to clean up the data sets.
Appendix C. Maintaining the persistent data store

The Tivoli Enterprise Portal displays two kinds of history data: short-term and long-term. Short-term history data, up to 24 hours worth, is retrieved from local storage on the host of the monitoring agent or the monitoring server to which it reports. Any data older than 24 hours is retrieved from the Tivoli Data Warehouse. Short-term historical data collection must be enabled and configured if you want to view long-term historical data. Historical data collection and warehousing is configured and started from the Tivoli Enterprise Portal.

If you intend to collect historical data, you must make provisions to manage the collected data. Without additional action, on distributed computers, the history data files grow unchecked, using up valuable disk space. On z/OS, the data sets allocated to store historical data are maintained by the persistent data store, which empties and reuses the data set with the oldest data.

The topics in this appendix explain how the persistent data store works and documents the maintenance procedure and options.

- “Overview of the persistent data store on z/OS”
- “Automatic maintenance process” on page 186
- “Making archived data available” on page 188
- “Exporting and restoring persistent data” on page 190
- “Data record format of exported data” on page 190
- “Extracting persistent data store data to flat files” on page 193
- “Components of the persistent data store” on page 195
- “Operation of the persistent data store” on page 196
- “Command interface” on page 197

This topics in this section describe the procedures used to maintain the persistent data store.

Overview of the persistent data store on z/OS

The persistent data store is used for writing and retrieving historical data. The data written to the persistent data store is organized by tables (attribute groups), groups, and data sets. Each table is assigned to a group. Multiple tables can be assigned to each group, and each group can have one or more data sets assigned to it. Typically, three data sets are assigned to each group.

One of the data sets in each collection group is kept empty so that it can be quickly switched to if the active data set fills up. After this switch is made, the persistent data store looks to see if there are any empty data sets in the group. If not, the automatic maintenance process empties the data set with the oldest data.

During configuration using the Configuration Tool you allocate space for the data sets and specify what is to be done with historical data that is sitting in a data set that is being emptied. The default is to delete the data. Alternatively, you can configure maintenance to back up the data to tape or DASD, export the data to an external program for processing, or extract the data and write it in a viewable format.

Note that if the Tivoli Data Warehouse is used, especially with a 1-hour warehousing interval, there is little or no advantage to invoking the BACKUP, EXPORT, or EXTRACT functions for data collected by OMEGAMON XE monitoring agents. These functions are more relevant if warehousing is not enabled, or if you are maintaining historical data for OMEGAMON agents whose data is not displayed by the Tivoli Enterprise Portal.
No historical data collection takes place until collection is configured and started through the History Collection Configuration window of the Tivoli Enterprise Portal. In the window, you specify the attribute groups for which you want data to be collected, the interval for data collection, the location where you want the collected data to be stored (at the monitoring server or at the agent), and whether you want to store the data in the Tivoli Data Warehouse so it will be available for long-term history data reporting.

**Automatic maintenance process**

When a data set becomes full, the persistent data store selects an empty data set to make active. When the data set is active, the persistent data store checks to see if there are any more empty data sets. If there are no more empty data sets, maintenance is started on the oldest data set.

The maintenance process consists of two files that are generated and tailored by the Configuration Tool and invoked by the persistent data store.

- **KPDPROC1**
  
  KPDPROC1 is a procedure that is started by the automatic maintenance processing if one of the three maintenance functions is configured. Limited information is passed to this started task to drive a CLIST in a TSO environment. The Configuration Tool creates this file and puts it into the RKANSAMU library for each runtime environment that has a persistent data store component. This procedure must be copied to a system level procedure library so the command issued to start it can be found.

  The parameters passed to KPDPROC1 vary based on the version of the Configuration Tool and the persistent data store. This document assumes the latest version is installed. Three parameters are passed to the started task:

  - **HILEV**
    
    High-level qualifier for the runtime environment that configured this version of the persistent data store. It is obtained by extracting information from the DD statement that points to the persistent data store control files.

  - **LOWLEV**
    
    Low-level qualifier for the sample library. It currently contains the RKANSAMU field name.

  - **data set**
    
    Fully qualified name of the data set being maintained. It is possible to have a data set name that does not match the high-level qualifier specified in the first parameter.

- **KPDPROCC**

  KPDPROCC is the CLIST that is executed by the KPDPROC1 procedure. The CLIST has the task of obtaining all of the information required to perform the maintenance, saving the data, and initialize the data set so it can be used again. This procedure performs the following actions:

  - Backs up the data
  - Deletes the data set
  - Allocates a new data set with the same parameters as before
  - Makes the new data set available for reading and writing

Before launching the KPDPROC1 process, the persistent data store checks to see if either the BACKUP, EXPORT, or EXTRACT function has been specified. If neither function has been specified, then the data set is initialized in the persistent data store started task and KPDPROC1 is not executed.

The Configuration Tool allows you to pick the first seven characters of the maintenance procedure names. KPDPROC is the default. If you changed KPDPROC to some other than the default during configuration, the suffixes remain 1 and C.

The persistent data store procedure runs with the user ID of the person who installed the product.
The part of maintenance you control

Most of the persistent data store maintenance procedure is automatic and does not require your attention. Through the Configuration Tool, you specify the EXTRACT, BACKUP and EXPORT options by indicating a Y or N for each data set group. See “Command interface” on page 197 for descriptions of additional commands that are used primarily for maintenance.

- BACKUP makes an exact copy of the data set being maintained.
- EXPORT writes the data to a flat file in an internal format that can be used by external programs to post process the data. This file is also used for recovery purposes when the persistent data store detects potential problems with the data.
- EXTRACT writes the data to a flat file in human-readable form, which is suitable for loading into other DBMS systems. The EXTRACT option is not strictly considered a way to maintain the data. If EXTRACT is specified, the data is pulled out and externalized in EBCDIC form, but the extraction does not empty the data set. If EXTRACT is specified, the data is extracted immediately from a data set that is switched from an active state to a full state. The data set is not initialized until it becomes a candidate for being emptied, so the most current data becomes available in an extracted form without that data being lost.

If none of the options are specified, the data in the data set being maintained is erased.

You can indicate one of these backup options:
- Back up the data for each data set group.
- Back up the data to tape or to DASD for all data set groups.

Indicating data set backup to tape or to DASD

For all data set groups that you selected to back up, you must indicate whether you want to back up the data to tape or to DASD. This decision applies to all data sets.

- If you are backing up data sets to tape, use KPDPROCC as included in the product.
- If you are backing up data sets to DASD, follow this procedure to modify KPDPROCC:
  1. With any editor, open the procedure in the &rhilev.&rte.RKANSAM(KPDPROCC) member.
  2. Find the statement
     
     TAPE = N
     
     and change the N to Y.
  3. Save the procedure.
  4. Copy procedure KPDPROCC to your system procedure library.

Tip: If you edit the KPDPROCC member of &rhilev.&rte.RKANSAM and then upgrade or reconfigure the runtime environment, your modifications might be overwritten.

Naming the export data sets

When you choose to export data, you are requesting to write data to a sequential data set. The names of all exported data sets follow this format:

&rhilev.&rte.&dsnlolev.Annnnnnn

where:
- &rhilev is the high-level qualifier of all data sets in the persistent data store, as you specified in the Configuration Tool
- &rte is the mid-level qualifier of all data sets in the persistent data store, as you specified in the Configuration Tool
- &dsnlolev is the low-level qualifier of the data set names as set by the Configuration Tool
- A is a required character
Making archived data available

This topic shows you how to make data available to those products that use the after the data has been backed up to DASD or to tape.

To make the data available after backup, you restore a connection between an archived data set and the monitoring server.

When the automatic maintenance facility backs up a data set in the persistent data store, it performs the following actions:

- Disconnects the data set from the monitoring server
- Copies the data set to tape or DASD in a format readable by the monitoring server
- Deletes and reallocates the data set
- Reconnects the empty data set to the monitoring server

To view archived data from the product, you must ensure that the data set is stored on an accessible DASD volume and reconnect the data set to the monitoring server.

Data set naming conventions

When the maintenance facility backs up a data set, it uses the following format to name the data set:

&rhilev.&rte.&dsnlolev.&nnnnnnn

where:

- &rhilev is the high-level qualifier of all data sets in the persistent data store, as you specified during configuration
- &rte is the mid-level qualifier of all data sets in the persistent data store, as you specified during configuration
- &dsnlolev is the low-level qualifier of the data set names as set by the Configuration Tool
- B is a required character
- nnnnnnn is a sequential number

Prerequisites

Before you begin to restore the connection between the archived data set and the monitoring server, find the following information:

- The name of the archived data set that contains the data you want to view. Your system programmer can help you locate the name of the data set.
- The name of the persistent data store group that corresponds to the data you want to view.

Finding background information

You can use the Configuration Tool to find the name of the persistent data store group to which the archived data set belongs by following this procedure:

1. Stop the Tivoli Enterprise Monitoring Server if it is running.
2. Log on to a TSO session and invoke ISPF.
3. At the ISPF Primary Option menu, enter 6 in the Option field to access the TSO command mode.
4. At the TSO command prompt, enter this command to start the Configuration Tool:
   
   EX 'shilev.INSLIB'

5. From the Main Menu, select Configure products and then Select product to configure.
6. From the Product Selection Menu, select the product.
7. On the Runtime Environments (RTEs) panel, specify C to configure the runtime environment.
8. On the Configure product panel, select Configure persistent data store and then Modify and review data store specifications.
9. Locate the low-level qualifier of the data set you want to reconnect and note the corresponding group name.
10. Press F3 until you exit the Configuration Tool.

**Connecting the data set to the monitoring server**

To reconnect the archived data set to the monitoring server so you can view the data from the product, follow this procedure:

1. If the data set is located on tape, use a utility such as IEBGENR to copy the data set to a DASD volume that is accessible by the monitoring server.
2. Copy job KPDCOMMJ from &thilev.TKANSAM to &rhilev.&rte.RKANSAMU.
3. Access job &rhilev.&rte.RKANSAMU(KPDCOMMJ) with any editor.
4. Substitute site-specific values for the variables in the job, as described in the comments at the beginning of the job.
5. Locate the COMM ADDFILE statement near the bottom of the job and remove the comment character (*).
6. Submit KPDCOMMJ to restore the connection between the data set you specified and the monitoring server.
7. To verify that the job ran successfully, you can view a report in RKPDLOG that lists all the persistent data store data sets that are connected to the monitoring server. RKPDLOG is the ddname of a SYSOUT file allocated to the monitoring server.
   Locate the last ADDFILE statement in the log and examine the list of data sets that follows the statement. If the job ran successfully, the name of the data set you reconnected is in the list.

**Disconnecting the data set**

The data set that you connected to the monitoring server is not permanently connected. The connection is removed automatically the next time the monitoring server stops. If you want to remove the data set from the persistent data store immediately after you view the data, follow this procedure:

1. Access job &rhilev.&rte.RKANSAMU(KPDCOMMJ) with any editor.
2. Retain all site-specific values that you entered when you modified the job to reconnect the data set in the previous procedure.
3. Locate the COMM ADDFILE statement near the bottom of the job and perform the following steps:
   a. Remove the comment character from the statement, if one exists.
   b. Change the word ADDFILE to the word DELFILE.
   c. Remove the Group parameter together with its value.
   d. Remove the RO parameter if it exists.
4. Submit KPDCOMMJ to remove the connection between the data set and the monitoring server.
   To verify that the job ran successfully, you can view a report in RKPDLOG that lists all data sets connected to the monitoring server.
   Locate the last DELFILE statement in the log and examine the list of data sets that follows the statement. If the job ran successfully, the name of the data set you disconnected is not included in the list.
5. If the data set is located on tape, you might want to conserve space by deleting the data set from DASD.
Exporting and restoring persistent data

In addition to the standard maintenance jobs used by the persistent data store, there are sample jobs distributed with the monitoring server that you can use to export data to a sequential file and then restore the data to the original indexed format.

These jobs are not tailored by the Configuration Tool at installation time and must be modified to add pertinent information.

Exporting persistent data

Follow this procedure to export persistent data to a sequential file:

1. Stop the monitoring server if it is running.
2. Copy &rhilev.&rte.RKANSAMU(KPDEXPTJ).
3. Update the JCL with the following values:
   
   &rhilev  high-level qualifier of the runtime environment where the persistent data store resides.
   &pdsn    fully qualified name of the persistent data store data set to be exported
   &expdsn  fully qualified name of the export file you are creating
   &unit2   DASD unit identifier for &expdsn
   &ssz     record length of output file (You can use the same record length as defined for &pdsn.)
   &sct     count of blocks to allocate (You can use the same size as the blocks allocated for &pdsn.)
   &bsz     &ssz value plus eight

   With the exception of &pdsn, these values can be found in the PDSLOG SYSOUT of the monitoring server started task.
4. Submit the job.

Restoring exported data

Follow this procedure to restore a previously exported persistent data store data set.

1. Copy &rhilev.&rte.RKANSAMU(KPDRESTJ).
2. Update the JCL with the following values:

   &rhilev  high-level qualifier of the runtime environment where the persistent data store resides
   &pdsn    fully qualified name of the persistent data store data set to be restored
   &expdsn  fully qualified name of the file you are creating
   &unit2   DASD unit identifier for &expdsn
   &group   identifier for the group that the data set will belong to
   &siz     size of the data set to be allocated, in megabytes

   With the exception of &pdsn, these values can be found in the PDSLOG SYSOUT of the monitoring server started task.
3. Submit the job.

Data record format of exported data

This section describes the format of the dictionary entries but not its contents. The meaning of the tables and columns is product-specific.
A single dictionary data set contains a description and mapping information for every table recorded in the original data set. In many cases, the tables have variable-length columns and some rows of data where columns are not available. The information about missing columns and lengths for variable columns is embedded in the data records. Some tables have columns that physically overlay each other. This must be taken into account when trying to obtain data for these overlays.

Data in the exported file is kept in internal format, and many of the fields are binary. The output file is made up of three sections with one or more data rows in each.

- Section 1 describes general information about the data source used to create the exported data.
- Section 2 contains a dictionary required for mapping the data.
- Section 3 contains the actual data rows.

The historical data is maintained in relational tables. Therefore, the dictionary mappings provide table and column information for every table that had data recorded for it in the persistent data store.

### Section 1 record

The Section 1 record is not required for mapping the data in the exported file. However, it is useful for determining how to reallocate a data set when a persistent data store file must be reconstructed.

Section 1 contains a single data row used to describe information about the source of the data recorded in the export file. The data layout for the record follows.

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecID</td>
<td>0</td>
<td>4</td>
<td>Char</td>
<td>Record ID. Contains AA10 for header record 1.</td>
</tr>
<tr>
<td>Length</td>
<td>4</td>
<td>4</td>
<td>Binary</td>
<td>Contains the record length of the header record.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>8</td>
<td>16</td>
<td>Char</td>
<td>Timestamp of export. Format: CYYMMDDHHMMSSMMM</td>
</tr>
<tr>
<td>Group</td>
<td>24</td>
<td>8</td>
<td>Char</td>
<td>Group name to which the data belongs.</td>
</tr>
<tr>
<td>Data Store Ver</td>
<td>32</td>
<td>8</td>
<td>Char</td>
<td>Version of KPDMANE used to record original data.</td>
</tr>
<tr>
<td>Export Version</td>
<td>40</td>
<td>8</td>
<td>Char</td>
<td>Version of KPDARCH used to create exported file.</td>
</tr>
<tr>
<td>Total Slots</td>
<td>48</td>
<td>4</td>
<td>Binary</td>
<td>Number of blocks allocated in original data set.</td>
</tr>
<tr>
<td>Used Slots</td>
<td>52</td>
<td>4</td>
<td>Binary</td>
<td>Number of used blocks at time of export.</td>
</tr>
<tr>
<td>Slot Size</td>
<td>56</td>
<td>4</td>
<td>Binary</td>
<td>Block size of original data set.</td>
</tr>
<tr>
<td>Expansion Area</td>
<td>60</td>
<td>20</td>
<td>---</td>
<td>Unused area.</td>
</tr>
<tr>
<td>Data Store Path</td>
<td>80</td>
<td>256</td>
<td>Char</td>
<td>Name of originating data set.</td>
</tr>
<tr>
<td>Export Path</td>
<td>336</td>
<td>256</td>
<td>Char</td>
<td>Name of exported data set.</td>
</tr>
</tbody>
</table>

### Section 2 records

Section 2 provides information about the tables and columns that are represented in Section 3. This section has a header record followed by a number of table and column description records.

### Dictionary header record

This is the first Section 2 record (and therefore the second record in the data set). It provides general information about the format of the dictionary records that follow. It is used to describe how many tables are defined in the dictionary section. The data layout for the dictionary header record follows.
Table 23. Section 2 data record format, exported persistent data

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecID</td>
<td>0</td>
<td>4</td>
<td>Char</td>
<td>Record ID. Contains DD10 for header record 2.</td>
</tr>
<tr>
<td>Dictionary Len</td>
<td>4</td>
<td>4</td>
<td>Binary</td>
<td>Contains the length of the entire dictionary.</td>
</tr>
<tr>
<td>Header Len</td>
<td>8</td>
<td>4</td>
<td>Binary</td>
<td>Length of the header record.</td>
</tr>
<tr>
<td>Table Count</td>
<td>12</td>
<td>4</td>
<td>Binary</td>
<td>Number of tables in dictionary (1 record per table).</td>
</tr>
<tr>
<td>Column Count</td>
<td>16</td>
<td>4</td>
<td>Binary</td>
<td>Total number of columns described.</td>
</tr>
<tr>
<td>Table Row Len</td>
<td>20</td>
<td>4</td>
<td>Binary</td>
<td>Size of table row.</td>
</tr>
<tr>
<td>Col Row Len</td>
<td>24</td>
<td>4</td>
<td>Binary</td>
<td>Size of column row.</td>
</tr>
<tr>
<td>Expansion</td>
<td>28</td>
<td>28</td>
<td>---</td>
<td>Unused area.</td>
</tr>
</tbody>
</table>

Table description record

Each table in the exported data set has a table record that provides its name and identifier, and additional information about the columns. All table records are provided before the first column record. The column records and all of the data records in section 3 use the identifier number to associate it with the appropriate table.

The map length and variable column count fields can be used to determine exactly where the data for each column starts and to determine if the column exists in a record. The format of the table description record is described in the table that follows.

Table 24. Section 2 table description record, exported persistent data

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecID</td>
<td>0</td>
<td>4</td>
<td>Char</td>
<td>Record ID. Contains DD20 for table record.</td>
</tr>
<tr>
<td>Identifier Num</td>
<td>4</td>
<td>4</td>
<td>Binary</td>
<td>Unique number for this table.</td>
</tr>
<tr>
<td>Application</td>
<td>8</td>
<td>8</td>
<td>Char</td>
<td>Application name table belongs to.</td>
</tr>
<tr>
<td>Table Name</td>
<td>16</td>
<td>10</td>
<td>Char</td>
<td>Table name.</td>
</tr>
<tr>
<td>Table Version</td>
<td>26</td>
<td>8</td>
<td>Char</td>
<td>Table version.</td>
</tr>
<tr>
<td>Map Length</td>
<td>34</td>
<td>2</td>
<td>Binary</td>
<td>Length of the mapping area.</td>
</tr>
<tr>
<td>Column Count</td>
<td>16</td>
<td>4</td>
<td>Binary</td>
<td>Count of columns in the table.</td>
</tr>
<tr>
<td>Variable Cols</td>
<td>36</td>
<td>4</td>
<td>Binary</td>
<td>Count of variable name columns.</td>
</tr>
<tr>
<td>Row Count</td>
<td>40</td>
<td>4</td>
<td>Binary</td>
<td>Number of rows in exported file for this table.</td>
</tr>
<tr>
<td>Oldest Row</td>
<td>44</td>
<td>16</td>
<td>Char</td>
<td>Timestamp for oldest row written for this table.</td>
</tr>
<tr>
<td>Newest Row</td>
<td>64</td>
<td>16</td>
<td>Char</td>
<td>Timestamp for newest row written for this table.</td>
</tr>
<tr>
<td>Expansion</td>
<td>80</td>
<td>16</td>
<td>---</td>
<td>Unused area.</td>
</tr>
</tbody>
</table>

Column description record

One record exists for every column in the associated table record. Each record provides the column name, type, and other characteristics. The order of the column rows is the same order in which the columns are displayed in the output row. However, some columns might be missing on any given row. Use the mapping structure defined under section 3 to determine whether a column is present.

The format of the column records follows.

Table 25. Section 2 column description record, exported persistent data

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecID</td>
<td>0</td>
<td>4</td>
<td>Char</td>
<td>Record ID. Contains DD30 for table record.</td>
</tr>
</tbody>
</table>
Table 25. Section 2 column description record, exported persistent data (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Ident</td>
<td>4</td>
<td>4</td>
<td>Binary</td>
<td>Identifier for the table this column belongs to.</td>
</tr>
<tr>
<td>Column Name</td>
<td>8</td>
<td>10</td>
<td>Char</td>
<td>Column name.</td>
</tr>
<tr>
<td>SQL Type</td>
<td>18</td>
<td>2</td>
<td>Char</td>
<td>SQL type for column.</td>
</tr>
<tr>
<td>Column Length</td>
<td>20</td>
<td>4</td>
<td>Binary</td>
<td>Maximum length of this column's data.</td>
</tr>
<tr>
<td>Flag</td>
<td>24</td>
<td>1</td>
<td>Binary</td>
<td>Flag byte.</td>
</tr>
<tr>
<td>Spare</td>
<td>25</td>
<td>1</td>
<td>---</td>
<td>Unused.</td>
</tr>
<tr>
<td>Overlay Col ID</td>
<td>26</td>
<td>2</td>
<td>Char</td>
<td>Column number if this is an overlay.</td>
</tr>
<tr>
<td>Overlay Col Off</td>
<td>28</td>
<td>2</td>
<td>Char</td>
<td>Offset into row for start of overlay column.</td>
</tr>
<tr>
<td>Alignment</td>
<td>30</td>
<td>2</td>
<td>---</td>
<td>Unused.</td>
</tr>
<tr>
<td>Spare 1</td>
<td>32</td>
<td>8</td>
<td>---</td>
<td>Unused.</td>
</tr>
</tbody>
</table>

Section 3 records

Section 3 has one record for every row of every table that was in the original persistent data store data set being exported. Each row starts with a fixed portion, followed by the actual data associated with the row. The length of the column map can be obtained from the table record (DD20). Each bit in the map represents one column. A 0 for the bit position indicates that the column data is not present, while a 1 indicates that data exists in this row for the column. Immediately following the column map field is an unaligned set of 2-byte-length fields. One of these length fields exists for every variable-length column in the table. This mapping information must be used to find the starting location for any given column in the data structure. The actual data starts immediately after the last length field.

If dealing with overlay columns, use the column offset defined in the DD30 records to determine the starting location for this type of column. Do not worry about overlaid columns with extracted data. If you want to look at the actual content of an overlaid column, you can expand the data by reinserting any missing columns and expanding all variable length columns to the maximum length before doing the mapping.

The table that follows maps the fixed portion of the data.

Table 26. Section 3 record format, exported persistent data

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecID</td>
<td>0</td>
<td>4</td>
<td>Char</td>
<td>Record ID. Contains ROW1 for column record.</td>
</tr>
<tr>
<td>Table Ident</td>
<td>4</td>
<td>4</td>
<td>Binary</td>
<td>Identifier for the table this record belongs to.</td>
</tr>
<tr>
<td>Row Length</td>
<td>8</td>
<td>4</td>
<td>Binary</td>
<td>Total length of this row.</td>
</tr>
<tr>
<td>Data Offset</td>
<td>12</td>
<td>4</td>
<td>Binary</td>
<td>Offset to start of data.</td>
</tr>
<tr>
<td>Data Length</td>
<td>16</td>
<td>4</td>
<td>Binary</td>
<td>Length of data portion of row.</td>
</tr>
<tr>
<td>Column Map</td>
<td>20</td>
<td>Varies</td>
<td>Binary</td>
<td>Column available map plus variable length fields.</td>
</tr>
</tbody>
</table>

Extracting persistent data store data to flat files

This section explains how to extract data from a persistent data store data set into a flat file in EBCDIC format. This information can be loaded into spreadsheets or databases.

The format of the data is converted to tab-delimited columns. The data is written to separate files for each table; therefore, the data format for all rows in each data set is consistent. The program also generates a separate file. This file contains a single row that provides the column names in the order in which the data
is organized. This file is also delimited for ease of use. An option (NOFF) on the KPDXTTRA program bypasses creating the separate file and places the column information as the first record of the data file.

This job is not tailored by the Configuration Tool at installation time and must be modified to add pertinent information.

The output from this job is written to files with the following naming standard:

```
&hilev.&rte.tablename.D
&hilev.&rte.tablename.H
```

The `tablename` is the first eight bytes of the actual table with any invalid characters changed to "#". The "D" suffix is for the data file and the "H" suffix is for the header file. Note that the NOFF parameter can be passed to the KPDXTTRA program to merge the data from the header file into the data file and therefore not create a unique header file. The high level and mid level qualifiers are passed as a parameter to the KPDXTTRA program (PREF=xxx) and are therefore controlled by the KPDPROCC member in the RKANSAMU data set.

If this job is run more than once on a given day, data is appended to any data previously extracted for that day.

All data sets are kept in read/write state even if they are not active. This makes the data sets unavailable if the monitoring server is running. Jobs cannot be run against the active data sets and the inactive data sets must be taken offline.

You can remove a data set from the monitoring server by issuing the modify command:

```
F stcname,KPDCMD QUIESCE FILE=DSN:data set
```

If you must run a utility program against an active data store, issue a SWITCH command prior to issuing this QUIESCE command.

### Extracting persistent data store data to EBCDIC files

Use this job to extract persistent data store data to EBCDIC files.

1. Copy `&hilev.&rte.RKANSAMU(KPDXTRAJ)`.
2. Update the JCL with the following values:
   ```
   &hilev High-level qualifier of the runtime environment where the persistent data store resides.
   &pdsn Fully qualified name of the persistent data store data set to be extracted
   &pref High-level qualifier for the extracted data
   ```
3. Add the parameters you want to use for this job
   ```
   PREF=
   IDENTIFIES the high-level qualifier for the output file. This field is required.
   
   DELIM=nn
   IDENTIFIES the separator character to be placed between columns. The default is 05.
   
   NOFF=
   IF used, causes the format file not to be generated. The column names are placed into the data file as the first record.
   
   QUOTES
   PLACES quotation marks around character type of data
   ```
4. Submit the job.
Format of extracted header and data file records

Header record
This section includes examples of header and data record formats.

The following example shows an extracted header file record:

```
TMZDIFF(int,0,4) WRITETIME(char,1,16) ORIGINNODE(char,2,128) QMNAME(char,3,48) APPLID(char,4,12)
APPLTYPE(int,5,4) SDATE_TIME(char,6,16) HOST_NAME(char,7,48) CNTTRANPGM(int,8,4) MSGSPUT(int,9,4)
MSGSREAD(int,10,4) MSGSBROWSD(int,11,4) OUTSIZEAVG(int,12,4) AVGMQTIME(int,14,4)
AVGAPPTIME(int,15,4) COUNTOFQS(int,16,4) AVGMQTINTM(int,18,4) DEFSTATE(int,19,4)
INT_TIME(int,20,4) INT_TIMEC(char,21,8) CNTTASKID(int,22,4) SAMPLES(int,23,4) INTERVAL(int,24,4)
```

Each field is separated by a tab character (by default). The data consists of the column name with a type, column number, and column length field within the parentheses for each column. The information within parentheses is used primarily to describe the internal formatting information, and therefore can be ignored.

Data record
The following example shows an extracted data file record for the preceding header file record:

```
0 "1000104003057000" "MQM7:SYSG:MQESA" "MQM7" "XCXS2DPL" 2 "1000104003057434"
"SYSG"1 0 0 0 0 0 2 90007 0 2 0 1 96856 "016: 01" 1 1 900
```

The header file and the data file match up as follows:

```
TMZDIFF 0 Integer
WRITETIME "1000104003057000" "Character
ORIGINNODE "MQM7:SYSG:MQESA" "Character
QMNAME "MQM7" "Character
... ... ...
SAMPLES 1 Integer
INTERVAL 900 Integer
```

Components of the persistent data store
The following sections discuss the components of the persistent data store.

**KPDMANE**
This is the primary executable program. It is a server for other applications running in the same address space. This program is designed to run inside the TMS:Engine address space as a separate subtask. Although it is capable of running inside the TMS:Engine, it does not make any use of TMS:Engine services. This is because the KPDMANE program is also used in other utility programs that are intended to run in batch mode. This is the program that eventually starts the maintenance task when it does a switch and determines that no empty data sets are available.

**KPDUTIL**
This program is used primarily to initialize one or more data sets for persistent data store use. The program attaches a subtask and starts the KPDMANE program in it. The DD statements used when this program is run dictate what control files are executed by the KPDMANE program.

**KPDARCH**
This client program pulls data from the specified data set and writes it out to a flat file. The program attaches a subtask and starts the KPDMANE program in it. The output data is still in an internal format, with index information excluded.

**KPDREST**
This client program reads data created by the KPDARCH program, inserts it into a data set in the
correct format for use by the persistent data store, and rebuilds the index information. The program attaches a subtask and starts the KPDMANE program in it.

**KPDXTRA**

This client program pulls data from a data set and writes it to one or more flat files, with all column data converted to EBCDIC and separated by tabs. This extracted data can be loaded into a database management system or into spreadsheet programs such as Excel. The program attaches a subtask and starts the KPDMANE program in it.

**KPDDSCO**

This program communicates with the started task that is running the persistent data store and sends it commands. The typical command is **RESUME**, which tells the persistent data store that it can resume using a data set. The program can use two forms of communication:

- SNA protocol to connect to the monitoring server and submit command requests.
- SVC 34 to issue a modify command to the started task.

This program also logs information in a general log maintained in the persistent data store tables.

---

### Operation of the persistent data store

The KPDMANE program invokes maintenance automatically in two places:

- On startup, while reading and processing data sets, KPDMANE examines internal data to determine whether each data set is in a known and stable state. If not, KPDMANE issues a **RECOVER** command.
- When KPDMANE is recording information from applications onto an active data set for a group, if it detects that it is running out of room, it executes the **SWITCH** command internally.

#### RECOVER logic

This code quiesces the data set and closes the file. Information is set up to request an **ARCHIVE**, **INIT**, and **RESTORE** operation to be performed by the maintenance procedures. An SVC 34 is issued for a **START** command on KPDPROC1 (or its overridden name). The command exits to the caller, and the data set is unusable until a **RESUME** command is executed.

#### SWITCH logic

The **SWITCH** command looks at all data sets assigned to the group and attempts to find an empty one. If no empty data sets are available, future attempts to write data to any data set in the group will fail. Normally, an empty data set is found and is marked as the active data set.

When a data set is deactivated because it is full, it is tested to see whether the **EXTRACT** option was specified. If so, the **EXTRACT** command for the data set is executed.

The next test is to check whether there are any empty data sets in the current group. If not, the code finds the data set with the oldest data and marks it for maintenance. With the latest release of the persistent data store, the code checks to see whether any of the maintenance options **BACKUP**, **EXPORT**, or **EXTRACT** were specified for this data set. If not, the **INITDS** command is executed. Otherwise, the **BACKUP** command is executed.

#### BACKUP logic

This code quiesces and closes the data set. A test is made to see whether either **BACKUP** or **EXPORT** is specified for the data set and appropriate options are set for the started task. The options always include a request to initialize the data set. An SVC 34 is issued to start the KPDPROC1 procedure. The code returns to the caller, and the data set is unavailable until the **RESUME** command is executed.

#### EXTRACT logic

This is similar to the **BACKUP** logic, except that the only option specified is an **EXTRACT** run with no initialization performed on the data set.
RESUME logic
This code opens the specified data set name and verifies that it is valid. The data set is taken out of the
quiesce state and made available for activation during the next SWITCH operation.

Command interface
The persistent data store uses a command interface to perform many of the tasks required for maintaining
the data sets used for historical data. Most of these commands can be invoked externally through a
command interface supported in the Engine environment. These commands can be executed by using the
standard z/OS MODIFY interface with the following format:

F stcname,KPDCMD command arguments

where

stcname Is the started task name of address space where the persistent data store is running.

command Is one of the supported dynamic commands.

arguments Are valid arguments to the specified command.

Commands
Many commands are supported by the persistent data store. The commands described in this section are
used primarily for maintenance.

SWITCH command
This dynamic command causes a data store file switch for a specific file group. At any given time,
update-type operations against tables in a particular group are directed to one and only one of the files in
the group. That one file is called the “active” file. A file switch changes the active file for a group. In other
words, the switch causes a file other than the currently active one to become the new active file.

If the group specified by this command has only one file, or the group currently has no inactive file that is
eligible for output, the switch is not performed.

At the conclusion of a switch, persistent data store starts the maintenance process for a file in the group if
no empty files remain in the group.

The [NO]EXTRACT keyword can be used to force or suppress an extract job for the data store file
deactivated by the switch.

Syntax:

SWITCH GROUP=groupid [ EXTRACT | NOEXTRACT ]

where

groupid Specifies the ID of the file group that is to be switched. The group must have multiple files
assigned to it.

EXTRACT Specifies that the deactivated data store file is to be extracted, even if the file’s GROUP
statement did not request extraction.

NOEXTRACT Specifies that extraction is not to be performed for the deactivated data store file. This
option overrides the EXTRACT keyword of the GROUP statement.

Note that if neither EXTRACT nor NOEXTRACT is specified, the presence or absence of the EXTRACT
keyword on the file’s GROUP statement determines whether extraction is performed as part of the switch.
BACKUP command
This command causes a maintenance task to be started for the data store file named on the command. The maintenance task typically deletes, allocates and initializes a data store file, optionally backing up or exporting the file before deleting it. (The optional export and backup steps are requested through parameters on the data store file's GROUP command in the RKPDIN file.)

Syntax:
BACKUP FILE=DSN:dsname

where

dsname: Specifies the physical data set name of the file that is to be maintained.

ADDFILE command
This command is used to dynamically assign a new physical data store file to an existing file group. The command can be issued any time after the persistent data store initialization has completed in the Tivoli Enterprise Monitoring Server. It can be used to increase the number of files assigned to a group or to bring old data back online. It cannot, however, be used to define a new file group ID. It can be used to add files only to groups that already exist as the result of GROUP commands in the RKPDIN input file.

Syntax:
ADDFILE GROUP=groupid FILE=DSN:dsname [ RO ] [ BACKUP ] [ ARCHIVE ]

where

groupid: Specifies the unique group ID of the file group to which a file is to be added.
dsname: Specifies the fully qualified name (no quotation marks) of the physical data set that is to be added to the group specified by groupid.
RO: Specifies that the file is to be read-only (that is, that no new data can be recorded to it). By default, files are not read-only (that is, they are modifiable). This parameter can also be specified as READONLY.
BACKUP: Specifies that the file is to be copied to disk or tape before being reallocated by the automatic maintenance task. (Whether the copy is to disk or tape is a maintenance process customization option.) By default, files are not backed up during maintenance.
ARCHIVE: Specifies that the file is to be exported before being reallocated by the automatic maintenance task. By default, files are not exported during maintenance.

DELFILE command
This command is used to drop one physical data store file from a file group's queue of files. It can be issued any time after persistent data store initialization has completed in the Tivoli Enterprise Monitoring Server.

The file to be dropped must be full, partially full, or empty; it cannot be the “active” (output) file for its group (if it is, the DELFILE command is rejected).

The DELFILE command is conceptually the opposite of the ADDFILE command, and is intended to be used to manually drop a file that was originally introduced by a GROUP or ADDFILE command. After a file has been dropped by DELFILE, it is no longer allocated to the Tivoli Enterprise Monitoring Server task and can be allocated by other tasks. Note that DELFILE does not physically delete a file or alter it in any way. To physically delete and uncatalog a file, use the REMOVE command.

Syntax:
DELFILE FILE=DSN:dsname
where

dname: Specifies the fully qualified (without quotation marks) name of the file that is to be dropped.

**EXTRACT command**
This command causes an extract job to be started for the data store file named on the command. The job converts the table data in the data store file to delimited text format in new files, and then signals the originating Tivoli Enterprise Monitoring Server to resume use of the data store file.

For each table extracted from the data store file, two new files are created. One file contains the converted data and one file contains a record describing the format of each row in the first file.

Syntax:
```ex
EXTRACT FILE=DSN:dname
```

where

dname: Specifies the physical data set name of the file to have its data extracted.

**INITDS command**
This command forces a data store file to be initialized in the address space where the persistent data store is running.

Syntax:
```ex
INITDS FILE DSN:dname
```

where

dname: Identifies the data set name of the data store file to be initialized.

**RECOVER command**
This command causes a recovery task to be started for the data store file named on the command. The recovery task attempts to repair a corrupted data store file by exporting it, reallocating and initializing it, and restoring it. The restore operation rebuilds the index information, the data most likely to be corrupted in a damaged file. The recovery is not guaranteed to be successful, however; some severe forms of data corruption are unrecoverable.

Syntax:
```ex
RECOVER FILE=DSN:dname
```

where

dname: Specifies the physical name of the data set to be recovered.

**RESUME command**
The RESUME command is used to notify the persistent data store that it can once again make use of the data set specified in the arguments. The file identified must be one that was taken offline by the backup, recover, or extract commands.

Syntax:
```ex
RESUME FILE=DSN:dname
```

where

dname: Specifies the physical name of the data set to be brought online.
Other useful commands

**QUERY CONNECT command**
The QUERY CONNECT command displays a list of applications and tables that are currently defined in the persistent data store. The output of this command shows the application names, table names, total number of rows recorded for each table, the group the table belongs to, and the current data set that the data is being written to.

Syntax:
```
QUERY CONNECT <ACTIVE>
```
where

**ACTIVE** - Optional parameter that displays only those tables that are active. An active table is one that has been defined and assigned to an existing group, and the group has data sets assigned to it.

**QUERY DATASTORE command**
The QUERY DATASTORE command displays a list of data sets known to the persistent data store. For each data set, the total number of allocated blocks, the number of used blocks, the number of tables that have data recorded, the block size, and status are displayed.

Syntax:
```
QUERY DATASTORE <FILE=DSN:dbname>
```
where

**FILE** - Optional parameter that allows you to specify that you are only interested in the details for a single data set. When this option is used, the resulting display is changed to show information that is specific to the tables being recorded in the data set.

**COMMIT command**
This dynamic command flushes to disk all pending buffered data. For performance reasons, the persistent data store does not immediately write to disk every update to a persistent table. Updates are buffered in virtual storage. Eventually the buffered updates are flushed (written to disk) at an optimal time. However, this architecture makes it possible for persistent data store files to become corrupted (not valid) if the files are closed prematurely, before pending buffered updates have been flushed. Such premature closings might leave inconsistent information in the files.

The following known circumstances might cause corrupted files:
- Severe abnormal monitoring server terminations that prevent the persistent data store recovery routines from executing
- IPLs performed without first stopping the monitoring server

The COMMIT command is intended to limit the exposure to data store file corruption. Some applications automatically issue this command after inserting data.

Syntax:
```
COMMIT
```
Appendix D. Product codes

The following table lists many of the product codes associated with the Tivoli Management Services components and OMEGAMON XE products. Where a component or product has more than one code, the principal configuration code is shown in bold print. Throughout this publication, the variable \( pp \) represents the product code.

Table 27. Two-character product codes

<table>
<thead>
<tr>
<th>Product or component</th>
<th>Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Tool (ICAT):PARMGEN</td>
<td>CI, CN, DZ, IN, KC, Kl, PZ, RT, Z2</td>
</tr>
<tr>
<td>End-to-End response time component</td>
<td>ET</td>
</tr>
<tr>
<td>EPILOG</td>
<td>EB, EC, ED, EI, EO, EP</td>
</tr>
<tr>
<td>Monitoring agent on z/OS (common to all monitoring agents)</td>
<td>AG</td>
</tr>
<tr>
<td>NetView for z/OS</td>
<td>NA</td>
</tr>
<tr>
<td>OMNIMON base (includes the IBM Tivoli OMEGAMON enhanced 3270 user interface and OMEGAMON subsystem)</td>
<td>CA, CC, EB, IA, JI, OB</td>
</tr>
<tr>
<td>OMEGAMON XE for CICS on z/OS</td>
<td>BG, C2, C3, C5, CP, IA, IN, OC</td>
</tr>
<tr>
<td>OMEGAMON XE for CICS TG on z/OS</td>
<td>GW</td>
</tr>
<tr>
<td>OMEGAMON XE for IMS™ on z/OS</td>
<td>AT, DF, I2, I3, I5, IA, ID, IN, IP, ML, OI, OR, OS, RI</td>
</tr>
<tr>
<td>OMEGAMON XE for Mainframe Networks</td>
<td>N3, ON</td>
</tr>
<tr>
<td>OMEGAMON XE for Storage on z/OS</td>
<td>DF, RC, S3</td>
</tr>
<tr>
<td>OMEGAMON XE on z/OS</td>
<td>CG, CS, EA, IA, IN, M2, M3, M5, MH, MR, OE, OM, OS, PM, RA, WD, XD, XO</td>
</tr>
<tr>
<td>OMEGAMON XE on z/VM and Linux</td>
<td>VL</td>
</tr>
<tr>
<td>OMEGAMON z/OS Management Console</td>
<td>HL</td>
</tr>
<tr>
<td>OMEGAVIEW</td>
<td>MV, SD, WO</td>
</tr>
<tr>
<td>Summarization and Pruning agent</td>
<td>SY</td>
</tr>
<tr>
<td>Tivoli Enterprise Monitoring Server</td>
<td>AG, CO, DS, EF, FA, FW, GL, IB, IH, LC, LG, LI, LS, MA, MS, NS, NV, O4, OU, OX, PD, PS, PT, QM, SH, SM, SS, TN, TR, UI, UT, VI, VT</td>
</tr>
<tr>
<td>Tivoli Enterprise Portal desktop client</td>
<td>CJ</td>
</tr>
<tr>
<td>Tivoli Enterprise Portal Server</td>
<td>CQ, FW</td>
</tr>
<tr>
<td>Tivoli Enterprise Portal browser client</td>
<td>CW</td>
</tr>
<tr>
<td>TMS:Engine</td>
<td>BB, DC, DE, DF, DH, LB, LC, LD, LE, LF, LG, LH, LI, LS, LT, LU, LV, LX, SB</td>
</tr>
<tr>
<td>Warehouse Proxy agent</td>
<td>HD</td>
</tr>
</tbody>
</table>
Appendix E. Predefining and managing OMEGAMON started tasks

If you are using System Automation for z/OS, make sure the automation rules are defined to start and manage the OMEGAMON started tasks. System Automation comes with an OMEGAMON policy that predefines these rules. See the System Automation publications at http://www.ibm.com/software/tivoli/products/system-automation-zos/

The following table lists the OMEGAMON started tasks (default names have been used) and associated automation dependencies.

<table>
<thead>
<tr>
<th>PARMGEN Configuration Parameter</th>
<th>Default started task name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dependency on VTAM</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VTAM must be running and the major node must be active.</td>
<td>KM2_CLASSIC_STC</td>
<td>CANSM2RC IBM Tivoli OMEGAMON XE on z/OS (3270)</td>
</tr>
<tr>
<td>KM2_CUA_STC</td>
<td>CANSM2</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270)</td>
</tr>
<tr>
<td>KM2_CUA_STC</td>
<td>CANSOCx</td>
<td>IBM Tivoli OMEGAMON XE for CICS on z/OS (where n = 1 per 3270 pair)</td>
</tr>
<tr>
<td>K22_CCnn_CLASSIC_STC</td>
<td>CANSOlx</td>
<td>IBM Tivoli OMEGAMON XE for IMS on z/OS (where n = 1 per 3270 pair)</td>
</tr>
<tr>
<td>K22_CUA_STC</td>
<td>CANSI2</td>
<td>IBM Tivoli OMEGAMON XE for IMS on z/OS (3270)</td>
</tr>
<tr>
<td>GBL_DB2_KD2_CLASSIC_STC</td>
<td>CANSO2</td>
<td>IBM Tivoli OMEGAMON XE for DB2 Performance Expert/Performance Monitor on z/OS (3270)</td>
</tr>
<tr>
<td>KDS_TEMM_STC</td>
<td>CANSDSST</td>
<td>Tivoli Enterprise Monitoring Server</td>
</tr>
<tr>
<td>RTE_CANSETE_STC</td>
<td>CANSETE</td>
<td>OMEGAMON End-to-End</td>
</tr>
<tr>
<td>KM2_EPILOG_COLLECTOR_STC</td>
<td>CANSM2HI</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Historical Collector (Epilog)</td>
</tr>
<tr>
<td>KM2_EPILOG_ZOOM_STC</td>
<td>CANSM2EZ</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Epilog Zoom</td>
</tr>
<tr>
<td>KD5_CUA_STC</td>
<td>CANSD5</td>
<td>IBM Tivoli OMEGAMON XE for Storage on z/OS (3270)</td>
</tr>
<tr>
<td>KMV_CUA_STC</td>
<td>CANSMV</td>
<td>IBM Tivoli OMEGAMON DE on z/OS OMEGAVIEW (Agent)</td>
</tr>
<tr>
<td>KON_CUA_STC</td>
<td>CANSON</td>
<td>IBM Tivoli OMEGAMON XE for Mainframe Networks on z/OS (3270)</td>
</tr>
<tr>
<td><strong>Dependency on TCP/IP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP/IP must be started, ready to accept binds, and able to resolve DNS names; this automation usually requires OMPROUTE.</td>
<td>K0S_TEMM_STC</td>
<td>CANSDSST</td>
</tr>
<tr>
<td>KM2_CUA_STC</td>
<td>CANSM2</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270)</td>
</tr>
<tr>
<td>KC5_AGT_STC</td>
<td>CANSC5</td>
<td>IBM Tivoli OMEGAMON XE for CICS on z/OS (Agent)</td>
</tr>
<tr>
<td>KD5_AGT_STC</td>
<td>CANSD5</td>
<td>IBM Tivoli OMEGAMON XE for DB2 Performance Expert/Performance Monitor on z/OS (Agent)</td>
</tr>
<tr>
<td>K15_AGT_STC</td>
<td>CANSI5</td>
<td>IBM Tivoli OMEGAMON XE for IMS on z/OS (Agent)</td>
</tr>
</tbody>
</table>
Table 28. OMEGAMON started tasks and automation dependencies (continued)

<table>
<thead>
<tr>
<th>PARMGEN Configuration Parameter</th>
<th>Default started task name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KN3_AGT_STC</td>
<td>CANSN3</td>
<td>IBM Tivoli OMEGAMON XE for Mainframe Networks on z/OS (Agent)</td>
</tr>
<tr>
<td>KMQ_AGT_STC</td>
<td>CANSMQ</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere MQ Monitoring (Agent)</td>
</tr>
<tr>
<td>KM2_AGT_STC</td>
<td>CANSMC</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere MQ Configuration (Agent)</td>
</tr>
<tr>
<td>KQ1_AGT_STC</td>
<td>CANSQI</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere Message Broker (Agent)</td>
</tr>
<tr>
<td>KYN_AGT_STC</td>
<td>CANSYN</td>
<td>ITCAM for Application Diagnostics on z/OS (Agent)</td>
</tr>
<tr>
<td>KOB_TOM_STC</td>
<td>CANSTOM</td>
<td>IBM Tivoli OMEGAMON enhanced 3270 user interface</td>
</tr>
<tr>
<td>KGW_AGT_STC</td>
<td>CANSGW</td>
<td>IBM Tivoli OMEGAMON XE for CICS TG on z/OS (Agent)</td>
</tr>
</tbody>
</table>

Dependency on Tivoli Enterprise Monitoring Server (CANSDSST)

These tasks retry if they do not succeed in connecting to CANSDSST. The dependency is optional.

<table>
<thead>
<tr>
<th>Default started task name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANSC5</td>
<td>IBM Tivoli OMEGAMON XE for CICS on z/OS (Agent)</td>
</tr>
<tr>
<td>CANSD5</td>
<td>IBM Tivoli OMEGAMON XE for DB2 Performance Expert/Performance Monitor on z/OS (Agent)</td>
</tr>
<tr>
<td>CANSI5</td>
<td>IBM Tivoli OMEGAMON XE for IMS on z/OS (Agent)</td>
</tr>
<tr>
<td>CANSM2HI</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Historical Collector (Epilog)</td>
</tr>
<tr>
<td>CANSDF</td>
<td>IBM Tivoli OMEGAMON XE for Storage on z/OS (3270)</td>
</tr>
<tr>
<td>CANSMQ</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere MQ Monitoring (Agent)</td>
</tr>
<tr>
<td>CANSMC</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere MQ Configuration (Agent)</td>
</tr>
<tr>
<td>CANSQI</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere Message Broker (Agent)</td>
</tr>
<tr>
<td>CANSYN</td>
<td>ITCAM for Application Diagnostics on z/OS (Agent)</td>
</tr>
<tr>
<td>CANSM2</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270)</td>
</tr>
<tr>
<td>CANSN3</td>
<td>IBM Tivoli OMEGAMON XE for Mainframe Networks on z/OS (Agent)</td>
</tr>
<tr>
<td>CANSTOM</td>
<td>IBM Tivoli OMEGAMON enhanced 3270 user interface</td>
</tr>
<tr>
<td>CANSGW</td>
<td>IBM Tivoli OMEGAMON XE for CICS TG on z/OS (Agent)</td>
</tr>
</tbody>
</table>

No dependencies

<table>
<thead>
<tr>
<th>Default started task name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANSCN</td>
<td>OMEGAMON Subsystem</td>
</tr>
<tr>
<td>CANSM2CS</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Common Storage Area Analyzer</td>
</tr>
</tbody>
</table>

Note: CANSOCx must come up before the CICS regions, or the OMEG INIT fails.

Workload Manager (WLM) settings for the OMEGAMON tasks

The various started tasks created during the configuration process have varying levels of priority. Some started tasks, typically the data collecting monitoring agents, require a high priority to enable the ability to make requests for data.
The following table lists the OMEGAMON started tasks (default names have been used) and indicates the
priority level to use to allocate these tasks:

*Table 29. OMEGAMON started tasks by priority group*

<table>
<thead>
<tr>
<th>PARMGEN Configuration Parameter</th>
<th>Default started task name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group 1: Collector tasks</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTE_CANSCN_STC</td>
<td>CANSCN</td>
<td>OMEGAMON Subsystem</td>
</tr>
<tr>
<td>GBL_DB2_KD2_CLASSIC_STC</td>
<td>CANSD2</td>
<td>IBM Tivoli OMEGAMON XE for DB2 Performance Expert/Performance Monitor on z/OS (3270)</td>
</tr>
<tr>
<td>KM2_CUA_STC</td>
<td>CANSCx</td>
<td>IBM Tivoli OMEGAMON XE for CICS on z/OS (where n = 1 per 3270 pair)</td>
</tr>
<tr>
<td>KC2_CCnn_CLASSIC_STC</td>
<td>CANSOIxn</td>
<td>IBM Tivoli OMEGAMON XE for IMS on z/OS (where n = 1 per 3270 pair)</td>
</tr>
<tr>
<td>KON_CUA_STC</td>
<td>CANSN</td>
<td>IBM Tivoli OMEGAMON XE for Mainframe Networks on z/OS (3270)</td>
</tr>
<tr>
<td>KM2_EPILOG_COLLECTOR_STC</td>
<td>CANSM2HI</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Historical Collector (Epilog)</td>
</tr>
<tr>
<td>KM2_CLASSIC_STC</td>
<td>CANSM2RC</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270)</td>
</tr>
<tr>
<td>KM2_CUA_STC</td>
<td>CANSM2</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270)</td>
</tr>
<tr>
<td>KM2_CSA_ANALYZER_STC</td>
<td>CANSM2CS</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Common Storage Area Analyzer</td>
</tr>
<tr>
<td>RTE_CANSETE_STC</td>
<td>CANSETE</td>
<td>OMEGAMON End-to-End</td>
</tr>
<tr>
<td>KDF_CUA_STC</td>
<td>CANSDF</td>
<td>IBM Tivoli OMEGAMON XE for Storage on z/OS (3270)</td>
</tr>
<tr>
<td>KC2_CCnn_CLASSIC_STC</td>
<td>CANSOIxn (one for each IMS subsystem)</td>
<td>IBM Tivoli OMEGAMON XE for IMS on z/OS (where n = 1 per 3270 pair)</td>
</tr>
</tbody>
</table>

**Group 2: OMEGAMON XE tasks**

This group of tasks also need high priority. SYSSTC or STCHIGH is generally fine for these.

<table>
<thead>
<tr>
<th>Group 2: OMEGAMON XE tasks</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>KDS_TEMSS_STC</td>
<td>CANSDDSST</td>
<td>Tivoli Enterprise Monitoring Server</td>
</tr>
<tr>
<td>KCS_AGTT_STC</td>
<td>CANS3C5</td>
<td>IBM Tivoli OMEGAMON XE for CICS on z/OS (Agent)</td>
</tr>
<tr>
<td>KDS_AGTT_STC</td>
<td>CANS3D5</td>
<td>IBM Tivoli OMEGAMON XE for DB2 Performance Expert/Performance Monitor on z/OS (Agent)</td>
</tr>
<tr>
<td>KI5_AGTT_STC</td>
<td>CANS3I5</td>
<td>IBM Tivoli OMEGAMON XE for IMS on z/OS (Agent)</td>
</tr>
<tr>
<td>KN3_AGTT_STC</td>
<td>CANS3N3</td>
<td>IBM Tivoli OMEGAMON XE for Mainframe Networks on z/OS (Agent)</td>
</tr>
<tr>
<td>KMQ_AGTT_STC</td>
<td>CANSMQ</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere MQ Monitoring (Agent)</td>
</tr>
<tr>
<td>KQI_AGTT_STC</td>
<td>CANSQI</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere Message Broker (Agent)</td>
</tr>
<tr>
<td>KYN_AGTT_STC</td>
<td>CANSYN</td>
<td>ITCAM for Application Diagnostics on z/OS (Agent)</td>
</tr>
<tr>
<td>KMC_AGTT_STC</td>
<td>CANSMC</td>
<td>IBM Tivoli OMEGAMON XE for Messaging-WebSphere MQ Configuration (Agent)</td>
</tr>
<tr>
<td>KOB_TOM_STC</td>
<td>CANSTOM</td>
<td>IBM Tivoli OMEGAMON enhanced 3270 user interface</td>
</tr>
</tbody>
</table>
Table 29. OMEGAMON started tasks by priority group (continued)

<table>
<thead>
<tr>
<th>PARMGEN Configuration Parameter</th>
<th>Default started task name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KGW_AGT_STC</td>
<td>CANSGW</td>
<td>IBM Tivoli OMEGAMON XE for CICS TG on z/OS (Agent)</td>
</tr>
</tbody>
</table>

**Group 3: OMEGAMON II presentation tasks**

This group of tasks need lower priority because they display data but do not collect it.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default started task name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KI2_CUA_STC</td>
<td>CANSI2</td>
<td>IBM Tivoli OMEGAMON XE for IMS on z/OS (3270)</td>
</tr>
<tr>
<td>KC2_CCnn_CUA_STC</td>
<td>CANSC2n</td>
<td>IBM Tivoli OMEGAMON XE for CICS on z/OS (where n = 1 per 3270 pair)</td>
</tr>
<tr>
<td>KD2_CUA_STC</td>
<td>CANSD2</td>
<td>IBM Tivoli OMEGAMON XE for DB2 Performance Expert/Performance Monitor on z/OS (Agent)</td>
</tr>
<tr>
<td>KMV_CUA_STC</td>
<td>CANSMV</td>
<td>IBM Tivoli OMEGAMON DE on z/OS OMEGAVIEW (Agent)</td>
</tr>
<tr>
<td>KM2_HIST_DATA_INTERFACE_STC</td>
<td>CANSM2HD</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Historical data interface</td>
</tr>
<tr>
<td>KM2_EPILOG_ZOOM_STC</td>
<td>CANSM2EZ</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Epilog Zoom</td>
</tr>
</tbody>
</table>

**Group 4: Maintenance tasks for historical data sets**

This group of tasks can be low-priority because they are started only when needed and they run for a short time.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default started task name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KM2_HIST_PROC_PRIMARY_STC</td>
<td>CANSM2HP</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Historical maintenance (primary)</td>
</tr>
<tr>
<td>KM2_HIST_PROC_SECONDARY_STC</td>
<td>CANSM2HS</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Historical maintenance (secondary)</td>
</tr>
<tr>
<td>RTE_PDS_KPDPROC_PREFIX</td>
<td>KPDPROCn</td>
<td>Common IBM Tivoli Monitoring Persistent Datastore high-level qualifier prefix historical maintenance procedure (where n = 1)</td>
</tr>
<tr>
<td>KM2_HIST_BATCH_REPORTER_STC</td>
<td>CANSM2BA</td>
<td>IBM Tivoli OMEGAMON XE on z/OS (3270) Historical maintenance (batch reporter)</td>
</tr>
<tr>
<td>KDF_DFDSS_SLAVE_STC</td>
<td>KDFDSPR</td>
<td>IBM Tivoli OMEGAMON XE for Storage on z/OS (3270 DFDSS Slave)</td>
</tr>
</tbody>
</table>
Appendix F. Documentation library

This appendix contains information about the publications related to the OMEGAMON XE products and to IBM Tivoli Monitoring and the commonly shared components of Tivoli Management Services. These publications are listed in the following categories:

- “Shared OMEGAMON XE publications”
- “IBM Tivoli Monitoring library” on page 208
- “Related publications” on page 209


To find a list of new and changed publications, click What's new on the Welcome page of the IBM Tivoli Monitoring and OMEGAMON XE Information Center. To find publications from the previous version of a product, click Previous information centers on the Welcome page for the product.

Shared OMEGAMON XE publications

The following publications provide information common to the OMEGAMON XE products:

- **IBM Tivoli OMEGAMON XE Monitoring Agents on z/OS: Quick Start Guide**, GI11-8918
  Summarizes the installation and setup of an OMEGAMON XE monitoring agent on z/OS.

- **IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Common Planning and Configuration Guide**, SC23-9734
  Gives instructions for planning and configuration tasks common to the components of Tivoli Management Services on z/OS and the OMEGAMON XE monitoring agents on z/OS.

- **IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Upgrade Guide**, SC27-2500
  Gives instructions for complete and staged upgrades to the latest versions of the OMEGAMON XE products.

- **IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Common Parameter Reference**, SC14-7280
  Provides reference information on parameters used for setting up runtime environments and configuring hub and remote Tivoli Enterprise Monitoring Servers on z/OS.

- **IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: PARMGEN Reference**, SC22-5435
  Provides detailed instructions and common configuration scenarios for creating and maintaining runtime environments using the PARMGEN configuration method.

- **IBM Tivoli OMEGAMON XE and Tivoli Management Services on z/OS: Enhanced 3270 User Interface Guide**, SC22-5426
  Describes the features of the interface and provides operating instructions and reference material.

- **IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: End-to-End Response Time Feature Reference**, SC27-2303
  Provides instructions and reference information for the End-to-End Response Time Feature, which supplies response time data to several OMEGAMON XE products.

- **IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Reports for Tivoli Common Reporting**, SC27-2304
  Explains how to use the Tivoli Common Reporting tool to create reports from data displayed in the Tivoli Enterprise Portal and stored in the Tivoli Data Warehouse database.
IBM Tivoli Monitoring library

The following publications provide information about IBM Tivoli Monitoring and about the commonly shared components of Tivoli Management Services:

- *IBM Tivoli Monitoring: Quick Start Guide*, GI11-8058
  Introduces the components of IBM Tivoli Monitoring.

- *IBM Tivoli Monitoring: Installation and Setup Guide*, GC32-9407
  Provides instructions for installing and configuring IBM Tivoli Monitoring components on Windows, Linux, and UNIX systems.

- *IBM Tivoli Management Services on z/OS: Program Directory for IBM Tivoli Management Services on z/OS*, GI11-4105
  Gives instructions for the SMP/E installation of the Tivoli Management Services components on z/OS.

- *IBM Tivoli Management Services on z/OS: Configuring the Tivoli Enterprise Monitoring Server on z/OS*, SC27-2313
  Gives detailed instructions for using the Configuration Tool to configure Tivoli Enterprise Monitoring Server on z/OS systems. Includes scenarios for using batch mode to replicate monitoring environments across the z/OS enterprise. Also provides instructions for setting up security and for adding application support to a Tivoli Enterprise Monitoring Server on z/OS.

  Describes the support tasks and functions required for the Tivoli Enterprise Portal Server and clients, including Tivoli Enterprise Portal user administration.

  Gives instructions for several methods of ensuring the availability of the IBM Tivoli Monitoring components.

- Tivoli Enterprise Portal online help
  Provides context-sensitive reference information about all features and customization options of the Tivoli Enterprise Portal. Also gives instructions for using and administering the Tivoli Enterprise Portal.

  Complements the Tivoli Enterprise Portal online help. The guide provides hands-on lessons and detailed instructions for all Tivoli Enterprise Portal features.

- *IBM Tivoli Monitoring: Command Reference*, SC32-6045
  Provides detailed syntax and parameter information, as well as examples, for the commands you can use in IBM Tivoli Monitoring.

- *IBM Tivoli Monitoring: Troubleshooting Guide*, GC32-9458
  Provides information to help you troubleshoot problems with the software.

- *IBM Tivoli Monitoring: Messages*, SC23-7969
  Lists and explains messages generated by all IBM Tivoli Monitoring components and by z/OS-based Tivoli Management Services components (such as Tivoli Enterprise Monitoring Server on z/OS and TMS:Engine).

  Introduces you to the IBM Tivoli Universal Agent, an agent of IBM Tivoli Monitoring. The IBM Tivoli Universal Agent enables you to use the monitoring and automation capabilities of IBM Tivoli Monitoring to monitor any type of data you collect.
• *IBM Tivoli Universal Agent API and Command Programming Reference Guide*, SC32-9461
  
  Explains the procedures for implementing the IBM Tivoli Universal Agent APIs and provides descriptions, syntax, and return status codes for the API calls and command-line interface commands.

• *IBM Tivoli Monitoring: Agent Builder User’s Guide*, SC32-1921
  
  Explains how to use the Agent Builder for creating monitoring agents and their installation packages, and for adding functions to existing agents.

## Documentation for the base agents

If you purchased IBM Tivoli Monitoring as a product, you received a set of base monitoring agents as part of the product. If you purchased a monitoring agent product (for example, an OMEGAMON XE product) that includes the commonly shared components of Tivoli Management Services, you did not receive the base agents.

The following publications provide information about using the base agents.

• Operating system agents:
  
  
  
  
  

• Agentless operating system monitors:
  
  
  
  
  

• Warehouse agents:
  
  

• System P agents:
  
  
  
  

• Other base agents:
  

## Related publications

Other sources of documentation

You can also obtain technical documentation about IBM Tivoli Monitoring and the OMEGAMON XE monitoring agents from the following sources:

- **Integrated Service Management Library**
  
  
  Integrated Service Management Library is an online catalog that contains integration documentation and other downloadable product extensions and accelerators.

- **Redbooks**
  
  
  IBM Redbooks and Redpapers include information about products from platform and solution perspectives.

- **Technotes**
  
  Technotes provide the latest information about known product limitations and workarounds. You can find Technotes through the IBM Software Support website at [http://www.ibm.com/software/support](http://www.ibm.com/software/support).

- **Tivoli wikis on the IBM developerWorks® website**
  
  Tivoli Wiki Central at [http://www.ibm.com/developerworks/wikis/display/tivoli/Home](http://www.ibm.com/developerworks/wikis/display/tivoli/Home) is the home for interactive wikis that offer best practices and scenarios for using Tivoli products. The wikis contain white papers contributed by IBM employees, and content created by customers and business partners.

  Two of these wikis are of particular relevance:


Appendix G. Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain support:

**Online**
Go to the IBM Software Support site at [http://www.ibm.com/software/support/probsub.html](http://www.ibm.com/software/support/probsub.html) and follow the instructions.

**IBM Support Assistant**
The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. To install the ISA software, go to [http://www.ibm.com/software/support/isa](http://www.ibm.com/software/support/isa).

**Troubleshooting Guide**
For more information about resolving problems, see the product's Troubleshooting Guide.
Notices

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

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

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

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

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

© Copyright IBM Corp. 2008, 2012
Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

**Trademarks**

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.
Glossary

A

activity. One phase within a sequence of predefined steps called a policy that automate system responses to a situation that has fired (that is, become true).

administration mode. See workspace administration mode” on page 226.

Advanced Encryption Standard. An encryption algorithm for securing sensitive but unclassified material designed by the National Institute of Standards and Technology (NIST) of the U.S. Department of Commerce. AES is intended to be a more robust replacement for the Data Encryption Standard. The specification calls for a symmetric algorithm (in which the same key is used for both encryption and decryption), using block encryption of 128 bits and supporting key sizes of 128, 192 and 256 bits. The algorithm was required to offer security of a sufficient level to protect data for the next 20 to 30 years. It had to be easily implemented in hardware and software and had to offer good defenses against various attack techniques. AES has been published as Federal Information Processing Standard (FIPS) 197, which specifies the encryption algorithm that all sensitive, unclassified documents must use.

AES. See Advanced Encryption Standard.

affinity. A label that classifies objects by managed system.

agent. Software installed on systems you want to monitor that collects data about an operating system, subsystem, or application running on each such system. Because an executable file gathers information about a managed system, there is always a one-to-one correspondence between them. Also called a Tivoli Enterprise Monitoring Agent.

agentless monitor. An agentless monitor uses a standard API (such as SNMP or CIM) to identify and notify you of common problems with the operating system running on a remote computer. Thus, as their name implies, the agentless monitors can retrieve monitoring and performance data without requiring OS agents on the computers being monitored. The agentless monitors provide monitoring, data gathering, and event management for Windows, Linux, AIX, HP-UX, and Solaris systems.

agentless monitoring server. A computer that has one or more agentless monitors running on it. Each agentless monitoring server can support up to 10 active instances of the various types of agentless monitors, in any combination. Each instance can communicate with up to 100 remote nodes, which means a single agentless monitoring server can support as many as 1000 monitored systems.

alert. A warning message or other indication that appears at a console to indicate that something has occurred or is about to occur that may require intervention.

alert monitor. A monitoring agent that monitors and relays alert information to the monitoring server. Sources of alerts include message logs, system consoles, and network and system management products.

algorithm. A set of well-defined rules for the solution of a problem in a finite number of steps. For example, a full statement of an arithmetic procedure for evaluating \(\sin(x)\) to a stated precision.

API. See Application Programming Interface.

application. A software component or collection of software components that performs specific user-oriented work (a task) on a computer. Examples include payroll, inventory-management, and word-processing applications.

Application Programming Interface. A set of multiple subprograms and data structures and the rules for using them that enables application development via a particular language and, often, a particular operating environment. An API is a functional interface supplied by the operating system or by a separately licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

arithmetic expression. A statement containing any combination of values joined together by one or more arithmetic operators in such a way that the statement can be processed as a single numeric value.

arithmetic operator. A symbol representing a mathematical operation (addition, subtraction, multiplication, division, or exponentiation), such as +, -, *, /, or ^.

associate. The process of linking a situation with a Navigator item that enables a light to go on and a sound to play for an open event. Predefined situations are associated automatically, as are situations created or edited through the Navigator item pop-up menu. When you open the Situation editor from the toolbar, any situations you create cannot be associated with a Navigator item during this editing session. You need to close the Situation editor, and then open it again from the pop-up menu of the Navigator item with which the situation is associated.
attribute. (1) A system or application element being monitored by the monitoring agent, such as Disk Name and Disk Read/Writes Per Second. (2) A characteristic of a managed object; that is, a field in the data structure of a managed object or in the workspace associated with that managed object. (3) A field in an ODBC-compliant database.

attribute group. A set of related attributes that can be combined in a data view or a situation. When you open the view or start the situation, data samples of the selected attributes are retrieved. Each type of monitoring agent has its own set of attribute groups.

autonomous-mode monitoring agents. These monitoring agents run without communicating directly with a Tivoli Enterprise Monitoring Server. An autonomous agent can emit Simple Network Management Protocol (SNMP) traps and Event Integration Facility (EIF) events directly to an OMNibus ObjectServer for private situations (but not for enterprise situations) that have turned true.

B

bar chart. A chart consisting of several bars of equal width. The value of the dependent variable is indicated by the height of each bar.

baroc files. Basic Recorder of Objects in C files define event classes for a particular IBM Tivoli Enterprise Console server. Baroc files also validate event formats based on these event-class definitions.

browser client. The software installed with the Tivoli Enterprise Portal Server that is downloaded to your computer when you start the Tivoli Enterprise Portal in browser mode. The browser client runs under the control of a Web browser.

C

capacity planning. The process of determining the hardware and software configuration required to accommodate the anticipated workload on a system.

chart. A graphical view of data returned from a monitoring agent. A data point is plotted for each attribute chosen and, for bar and pie charts, a data series for each row. Types of charts include pie, bar, plot, and gauge.

CIM. See "Common Information Model."

class file. A file containing Java object code for a single Java object class.

class loader. A Java component that loads Java class files.

client. An application that receives requested data from a server.

client/server architecture. An architecture in which the client (usually a personal computer or workstation) is the machine requesting data or services and the server is the machine supplying them. Servers can be microcomputers, minicomputers, or mainframes. The client provides the user interface and may perform application processing. In IBM Tivoli Monitoring the Tivoli Enterprise Portal is the client to the Tivoli Enterprise Portal Server, whereas the portal server is the client to the Tivoli Enterprise Monitoring Server.

A database server maintains the databases and processes requests from the client to extract data from or to update the database. An application server provides additional business-support processing for the clients.

Common Information Model. An XML-based standard for defining device and application characteristics so that system administrators and management programs can monitor and control them using the same set of tools, regardless of their differing architectures. CIM provides a more comprehensive toolkit for such management functions than the Simple Network Management Protocol.

Common Object Request Broker Architecture. An industry specification for the design and standardization of different types of object request brokers (ORBs). ORBs allow different computers to exchange object data; CORBA enables ORBs from different software vendors (often running under dissimilar computer systems and operating systems) to exchange object data. CORBA facilitates communication among program components in a network using objects. The Tivoli Enterprise Portal Server is a CORBA implementation.

condition. An expression that evaluates to either true or false. It can be expressed in natural language text, in mathematically formal notation, or in a machine-readable language.

Configure History permission. Your userid must have Configure History permission to open the History Collection Configuration window for setting up history files and data rolloff. If you do not have this permission, you cannot see the menu item or tool for historical configuration.

Configuration Tool, z/OS (ICAT). A REXX-based tool for configuring OMEGAMON XE products running on zSeries systems, after they have been installed using the System Modification Program/Extended (SMP/E) tool.

CORBA. See "Common Object Request Broker Architecture."

critical state. The indication that a situation associated with a Navigator item is in an unacceptable state and that you must take corrective action. The critical state is represented by the color red.
Custom Navigator Views permission. Your userid has a Modify check box for the Custom Navigator Views feature. This permission must be enabled for you to open the Navigator view editor to maintain and update Navigator views.

Data Encryption Standard. A widely used method of private-key data encryption that originated at IBM in 1977 and was adopted by the U.S. Department of Defense. DES supports 72 quadrillion or more possible encryption keys; for each message, the key is chosen at random from among this enormous number of possible keys. Like all other private-key cryptographic methods, both the sender and the receiver must know and use the same private key.

DES applies a 56-bit key to each 64-bit block of data. Although this is considered strong encryption, many companies use triple DES, which applies three keys in succession.

data source name. The name that is stored in the database server and that enables you to retrieve information from the database through ODBC. The DSN includes such information as the database name, database driver, userid, and password.

data sources. Data pertaining to J2EE data sources, which are logical connections to database subsystems.

data warehouse. A central repository for all or significant parts of the data that an organization's business systems collect.

database. A collection of both interrelated and independent data items that are stored together on a computer disk to serve multiple applications.

DB2 for the workstation. IBM's DB2 Database for Linux, UNIX, and Windows systems is a relational database management system that runs on desktop computers. You install a DB2 database on the same system as the Tivoli Enterprise Portal Server; it stores the portal server's queries, customized workspaces, userids, and custom Navigator views. DB2 for the workstation can also serve as the data repository for the Tivoli Data Warehouse, which stores historical monitoring information.

default. Pertaining to an attribute, value, or option that is assumed when none is explicitly specified.

Demilitarized Zone. The area of a World Wide Web application that a company can use to host Internet services without allowing unauthorized access.

Derby. An open-source, public-domain, relational database management system implemented in Java and designed to conform to accepted database standards (such as SQL and JDBC). Derby came about when IBM contributed its Cloudscape database manager to the Apache project and features a small machine footprint. IBM Tivoli Monitoring implements Derby as an embedded database within its Tivoli Enterprise Portal Server; in other words, the database is installed with the portal server, and it runs within the portal server's Java virtual machine.

DES. See "Data Encryption Standard."

desktop client. Software supplied with IBM Tivoli Monitoring that you install on a workstation that you plan to use for interacting with the Tivoli Enterprise Portal Server and the Tivoli Enterprise Monitoring Server. The desktop Tivoli Enterprise Portal client provides the graphical user interface into the IBM Tivoli Monitoring network.

detailed attribute name. The name used in formulas, expert advice, Take Action commands, and headers and footers when referencing a monitoring agent attribute. In the Properties and Situation editors, you click Show Formula, and then check Show detailed formula to see the detailed attribute name.

display item. An attribute designated to further qualify a situation. With a display item set for a multiple-row attribute group, the situation continues to look at the other rows in the sample and opens more events if other rows qualify. The value displays in the event workspace and in the message log and situation event console views. You can select a display item when building a situation with a multiple-row attribute group.

distribution. The managed systems on which the situation is running.

DLL. See "Dynamic Link Library."

DMZ. See "Demilitarized Zone."

drill down. To access information by starting with a general category and moving through the hierarchy of information, for example, in a database, to move from file to record to field.

DSN. See "data source name."

Dynamic Link Library. A composite of one or more executable objects that is bound together by a linking procedure and loaded at run time (rather than when the application is linked). The code and data in a dynamic link library can be shared by several applications simultaneously. DLLs apply only to Windows operating environments.

E

EIB. See "Enterprise Information Base" on page 218.

EIF. See "Event Integration Facility" on page 218.
endcode. You assign endcodes in a policy when you connect one activity to another. The endcode indicates the result of this activity that triggers the next activity.

Enterprise Information Base. A database used by the Tivoli Enterprise Monitoring Server that serves as a repository of shared objects for all systems across your enterprise. The EIB stores all persistent data, including situations, policies, user definitions, and managed-object definitions.

temporary. A situation that is created for a Tivoli Enterprise Monitoring Agent that reports events to the Tivoli Enterprise Monitoring Server to which it connects. Enterprise situations are centrally defined at the monitoring server and distributed at agent startup. See also [situation] on page 223.

event. An action or some occurrence, such as running out of memory or completing a transaction, that can be detected by a situation. Events cause a change in the state of a managed system associated with a situation, thereby make the situation true and causing an alert to be issued.

event indicator. The colored icon that displays over a Navigator item when an event opens for a situation running on that item.

Event Integration Facility. An application programming interface that external applications can invoke to create, send, or receive events. These events are in the same format as Tivoli Enterprise Console events and are referred to as either EIF events or TEC/EIF events.

event item. A Navigator item that shows when you open the event workspace for a true situation (by selecting it from the event flyover listing or from the situation event console pop-up menu).

event sound. The sound file that plays when an event opens. This sound file is set in the Situation editor when the situation is associated with a Navigator item and can differ for different Navigator items.

expert advice. A description within the Situation editor of each situation provided with a monitoring agent to help you quickly understand and interpret events arising from it.

Extensible Markup Language. A data-description language derived from Standard Generalized Markup Language (SGML); also a tool for encoding messages so they describe their own fields. You use XML to format a document as a data structure. As program objects, such documents can have their contents and data hidden within the object, which allows you to control who can manipulate the document and how. Additionally, documents can carry with them the object-oriented procedures called methods. The XML standard aids in exchanging data between applications and users.

F

filter criteria. These criteria limit the amount of information returned to the data view in response to a query. You can apply a prefilter to the query to collect only certain data, or apply a postfilter to the view properties to show only certain data from the information collected.

fix pack. A tested collection of all cumulative maintenance for a product, up to the release of the fix pack. It can also contain fixes that have not been shipped previously, but it might contain no new function.

G

gurefereferenced map. A special type of graphic that has built-in knowledge of latitude and longitude and can be zoomed into and out of quickly. The Tivoli Enterprise Portal uses proprietary .IVL files generated with the map-rendering component. These files cannot be opened or saved in a graphics editor.

GSKit. The Global Security Toolkit provides SSL (Secure Sockets Layer) processing within protocols such as SPIPE and HTTPS. On z/OS systems, GSKit is known as the Integrated Cryptographic Service Facility, or ICSF.

H

hierarchical. Pertaining to data that is organized on computer systems using a hierarchy of containers, often called folders (that is, directories) and files. In this scheme, folders can contain other folders and files. The successive creation of folders within folders creates the levels of organization, which is the hierarchy.

historical collection. A definition for collecting and storing data samples for historical reporting. The historical collection identifies the attribute group, any row filtering you have assigned, the managed system distribution, frequency of data collection, where to store it for the short term, and whether to save data long term (usually to the Tivoli Data Warehouse).

historical data management. The procedures applied to short-term binary history files that roll off historical data to either the Tivoli Data Warehouse or to delimited text files (the karloff utility on UNIX or Windows systems; ddname KBDXTRA for the z/OS Persistent Datastore), and then delete entries in the short-term history files over 24 hours old, thereby making room for new entries.

hot standby. A redundant Tivoli Enterprise Monitoring Server that, if the primary or hub monitoring server should fail, assumes the responsibilities of the failed monitoring server.
HTTP. The Hypertext Transfer Protocol is a suite of Internet protocols that transfer and display hypertext documents within Web browsers.

HTTP sessions. Data related to invocations of specific World Wide Web sites.

HTTPS. The Secure Hypertext Transport Protocol is an implementation of the Hypertext Transfer Protocol (HTTP) that relies on either the Secure Sockets Layer (SSL) API or the Transport Layer Security (TLS) API to provide your users with secure access to your site's Web server. These APIs encrypt and then decrypt user page requests as well as the pages returned by the Web server.

hub Tivoli Enterprise Monitoring Server. (1) A central host system that collects the status of situations running on your systems. (2) The monitoring server that your site has selected to act as the focal point to which all portal servers and remote monitoring servers in this monitored network connect. A remote monitoring server passes its collected data to the hub to be made available to clients, creating an enterprise-wide view.

IBM Tivoli Monitoring. A client/server implementation for monitoring enterprise-wide computer networks that comprises a Tivoli Enterprise Monitoring Server, an application server known as the Tivoli Enterprise Portal Server, one or more Tivoli Enterprise Portal clients, and multiple monitoring agents that collect and distribute data to the monitoring server.

IIOP. See “Internet Inter-ORB Protocol.”

input data. Data provided to the computer for further processing. See also “output data” on page 221.

integral Web server. A proprietary Web server developed for IBM Tivoli Monitoring that is installed and configured automatically with the Tivoli Enterprise Portal Server. You enter the URL of the integral Web server to start the Tivoli Enterprise Portal client in browser mode.

Internet Inter-ORB Protocol. An Internet communications protocol that runs on distributed platforms. Using this protocol, software programs written in different programming languages and running on distributed platforms can communicate over the Internet. IIOP, a part of the CORBA standard, is based on the client/server computing model, in which a client program makes requests of a server program that waits to respond to client requests. With IIOP, you can write client programs that communicate with your site's existing server programs wherever they are located without having to understand anything about the server other than the service it performs and its address (called the Interoperable Object Reference, IOR, which comprises the server's port number and IP address).

Interoperable Object Reference. Connects clients to the Tivoli Enterprise Portal Server. The IOR identifies a remote object, including such information as name, capabilities, and how to contact it. The URL may include an IOR because it goes through the Web server; the portal server uses it to tell the client which IOR to fetch. After it does that, the portal server extracts the host and port information and tells the client where to route the request.

interval. The number of seconds that have elapsed between one sample and the next.

IOR. See “Interoperable Object Reference.”

J

Java Database Connectivity. A standard API that application developers use to access and update relational databases (RDBMSes) from within Java programs. The JDBC standard is based on the X/Open SQL Call Level Interface (CLI) and complies with the SQL-92 Entry Level standard; it provides a DBMS-independent interface that enables SQL-compliant database access for Java programmers.

Java Management Extensions. A set of Java classes for application and network management in J2EE environments. JMX provides Java programmers a set of native Java tools called MBeans (managed beans) that facilitate network, device, and application management. JMX provides a Java-based alternative to the Simple Network Management Protocol.

JDBC. See “Java Database Connectivity.”

JMX. See “Java Management Extensions.”

L


Lightweight Directory Access Protocol. A protocol that conforms to the International Standards Organization's X.500 directory standard that uses TCP/IP to access directory databases where applications can store and retrieve common naming and location data. For example, applications can use LDAP to access such directory information as email addresses, service configuration parameters, and public keys.

location broker. The component that manages connections for the hub monitoring server, enabling it to find all other Tivoli Management Services components, including remote monitoring servers, the Tivoli Enterprise Portal Server, and monitoring agents.
**M**

**managed object.** An icon created in the Tivoli Enterprise Portal from a managed object template that represents resources you monitor using situations. Managed objects are converted to items in the Navigator's Logical view.

**managed system.** A particular operating system, subsystem, or application in your enterprise where a monitoring agent is installed and running. A managed system is any system that IBM Tivoli Monitoring is monitoring.

**managed system group.** (Formerly managed system list.) A named, heterogeneous group of both similar and dissimilar managed systems organized for the distribution of historical collections, situations, and policies, and for assignment to queries and items in custom Navigator views. For example, you might create a managed system group named IT_London for a geographic region and another named Credit_Approval for a functional area of your organization.

If a managed system group is updated (usually when a constituent managed system is added or deleted), then all the historical collections, situations, and policies that use that group are redistributed to all managed systems in the group. Managed system groups are created, modified, or deleted either by the Tivoli Enterprise Portal's Object Group editor or by using the tacmd CLI command with the createsystemlist, editsystemlist, or deletesystemlist keywords; they are maintained by the Tivoli Enterprise Monitoring Server.

**MBeans.** Managed beans are Java objects that represent managed resources such as devices, services, and applications. The management functions are provided by the MBean server.

**Microsoft Management Console.** This feature of Microsoft's various Windows Server environments provides a centralized, consistent, and extensible interface to Windows' various monitoring and management utilities. In particular, MMC manages directory services, job scheduling, event logging, performance monitoring, and user environments.

**middleware.** Software that enables the exchange of information between components in a distributed computing environment. The middleware is the data-exchange and communications channel that allows programs to cooperate with each other without having to know details about how they are implemented or where they are deployed. Middleware typically provides a range of related facilities such as persistence, auditing, and the ability to build a transactional unit of work. IBM's CICS and WebSphere MQ are examples of middleware.

**method.** In object-oriented programming, the software that implements an object's behavior as specified by an operation.

**migrating.** Preserving your customized configuration data so you can use it again after installing a newer version of the product.

**MMC.** See Microsoft Management Console.

**monitor.** An entity that performs measurements to collect data pertaining to the performance, availability, reliability, or other attributes of applications or the systems on which those applications rely. These measurements can be compared to predefined thresholds. If a threshold is exceeded, administrators can be notified, or predefined automated responses can be performed.

**monitor interval.** A specified time, scalable to seconds, minutes, hours, or days, for how often the monitoring server checks to see if a situation has become true. The minimum monitor interval is 30 seconds; the default value is 15 minutes.

**monitoring.** Running a hardware or software tool to monitor the performance characteristics of a system.

**N**

**NAT.** See Network Address Translation.

**Navigator.** The upper-left pane of the Tivoli Enterprise Portal window. The Navigator Physical view shows your network enterprise as a physical hierarchy of systems grouped by platform. You can also define other views that create logical hierarchies grouped as you specify, such as by department or function.

**Network Address Translation.** A scheme used by local-area networks (LANs) to establish an internal and external set of IP addresses. Internal IP addresses are kept private and must be translated to and from the external addresses for outbound and inbound communications. NAT is often used in firewall configurations.

**Network File System.** A client/server file system developed by Sun Microsystems that, once mounted (that is, made accessible), allows a user on an NFS client to view, store, and update files on a remote computer (the NFS server) as though they were on the user's own computer. The portion of the mounted file system that each user can access and in what ways is determined by the user's own file-access privileges and restrictions.

Both the NFS server and client use TCP/IP's User Datagram Protocol as the mechanism for sending file contents and updates back and forth. NFS has been designated a file server standard: it uses the Remote Procedure Call method of communication between computers.
node. (1) In networking, a point capable of sending and receiving data. A node can be a device, such as a printer or workstation, a system, a storage location on a disk, or a single computer. (2) Any managed system, such as an AIX-based pSeries® server, that IBM Tivoli Monitoring is monitoring. A node can also be a managed system of subnodes, all of which are being managed as components of the primary node.

non-agent bundles. You can use these custom bundles to remotely deploy components that need not connect to a Tivoli Enterprise Monitoring Server, such as those that support other Tivoli products like IBM Tivoli Netcool/OMNIbus.

O

object. An instance of a class, which comprises an implementation and an interface. An object reflects its original, holding data and methods and responds to requests for services. CORBA defines an object as a combination of state and a set of methods characterized by the behavior of relevant requests.

ODBC. See "Open Database Connectivity."

OMEGAMON Dashboard Edition (OMEGAMON DE). The OMEGAMON implementation that includes all the features of the Tivoli Enterprise Portal provided with OMEGAMON XE, plus application-integration components that facilitate an enterprise-wide view of your computing environment. OMEGAMON DE’s workspaces integrate the data from multiple OMEGAMON Monitoring Agents into one network-wide view.

OMEGAMON Extended Edition (OMEGAMON XE). The IBM Tivoli Monitoring implementation of a single OMEGAMON Monitoring Agent. OMEGAMON XE displays the monitoring data from each OMEGAMON Monitoring Agent independently, without integrating it into the enterprise-wide workspaces provided by OMEGAMON DE.

OMEGAMON Monitoring Agent. The software process that probes a managed z/OS system or subsystem (such as CICS) for data. The monitoring agent sends that monitoring information back to the Tivoli Enterprise Monitoring Server and then on to the Tivoli Enterprise Portal Server to be formatted into table and chart views for display on a Tivoli Enterprise Portal client.

OMEGAMON Tivoli Event Adapter. Invokes the Event Integration Facility API to synchronize IBM Tivoli Monitoring events with the IBM Tivoli Enterprise Console product. OTEA is a component of the Tivoli Enterprise Monitoring Server; it forwards IBM Tivoli Monitoring events to Tivoli Enterprise Console and maps them to their corresponding Tivoli Enterprise Console event classes based on the situation name's suffix, either _Warning or _Critical.

Integrating these products requires two parts: a Tivoli Enterprise Monitoring Server piece (included with IBM Tivoli Monitoring version 6.1 and subsequent releases) called the OMEGAMON Tivoli Event Adapter, and a Tivoli Enterprise Console piece called the Situation Update Forwarder that is installed on the Tivoli Enterprise Console server.

Open Database Connectivity. A standard API for accessing data in both relational and nonrelational database systems using procedural, non-object-based languages such as C. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface.

Tivoli Enterprise Portal users can access the Query editor to write custom SQL queries for creating views that retrieve data from ODBC-compliant databases.

OTEA. See "OMEGAMON Tivoli Event Adapter."

output data. Data resulting from computer processing. See also input data on page 219.

parameter. A value or reference passed to a function, command, or program that serves as input or to control actions. The value is supplied by a user or by another program or process.

P

PDS. See "Persistent Datastore."

PerfMon. See "Performance Monitor."

performance. A major factor in measuring system productivity. Performance is determined by a combination of throughput, response time, and availability.

performance analysis. The use of one or more performance tools to investigate the reasons for performance improvement or deterioration.


Persistent Datastore. A set of z/OS data sets where IBM Tivoli Monitoring running on z/OS systems stores historical monitoring data.

platform. The operating system on which the managed system is running, such as z/OS or Linux. The Navigator’s Physical mapping places the platform level under the Enterprise level.
policy. A set of automated system processes that can perform actions, schedule work for users, or automate manual tasks, frequently in response to events. Policies are the IBM Tivoli Monitoring automation tool; they comprise a series of automated steps, called activities, whose order of execution you control.

In most cases, a policy links a Take Action command to a situation that has turned true. When started, the policy’s workflow progresses until all activities have been completed or until the Tivoli Enterprise Portal user manually stops the policy. You can create both policies that fully automate workflow strategies and those that require user intervention. As with situations, policies are distributed to the managed systems you want to monitor and to which you are sending commands.

private situation. A situation is defined in an XML-based private configuration file for the local Tivoli Enterprise Monitoring Agent or Tivoli System Monitor Agent and that does not interact with a Tivoli Enterprise Monitoring Server. Such events can be sent by using either EIF or SNMP alerts to a receiver such as IBM Tivoli Enterprise Console or Netcool/OMNibus. See also “situation” on page 223.

product code. The three-letter code used by IBM Tivoli Monitoring to identify the product component. For example, the product code for IBM Tivoli Monitoring for WebSphere Application Server is KWE.

Properties editor. A multi-tabbed window for specifying the properties of the individual views that make up a workspace, as well as the general workspace properties.

pure event. A pure event is one that occurs automatically, such as when a paper-out condition occurs on the printer or when a new log entry is written. Situations written to notify you of pure events remain true until they are manually closed or automatically closed by an UNTIL clause. See also “event” on page 218.

Q

query. A particular view of specified attributes of selected instances of a set of managed-object classes, arranged to satisfy a user request. Queries are written using SQL.

R

remote deployment. Using IBM Tivoli Monitoring software, you can deploy agents and other non-agent, Tivoli Management Services-based components to remote nodes without your having to sign onto those nodes and perform the installation and configuration steps yourself. Remote deployment requires two pieces on the destination node: (1) a bundle containing the component code and the instructions for installing and configuring it and (2) an operating-system agent to read the bundle and perform the installation and configuration steps.

Remote Procedure Call. A protocol based on the Open Software Foundation’s Distributed Computing Environment (DCE) that allows one program to request services from a program running on another computer in a network. RPC uses the client/server model: the requesting program is the client, and the responding program is the server. As with a local procedure call (also known as a function call or a subroutine call), an RPC is a synchronous operation: the requesting program is suspended until the remote procedure returns its results.

remote Tivoli Enterprise Monitoring Server. A remote monitoring server collects monitoring data from a subset of your site’s monitoring agents and passes its collected data to the hub Tivoli Enterprise Monitoring Server to be made available to one or more Tivoli Enterprise Portal clients through the Tivoli Enterprise Portal Server, thereby creating an enterprise-wide view.

rolloff. The transfer of monitoring data to a data warehouse.

RPC. See “Remote Procedure Call.”

RTE. See “runtime environment.”

runtime environment. A group of execution libraries that provide an operational environment on a z/OS system. RTEs execute OMEGAMON products on a z/OS image.

runtime libraries. Libraries in the runtime environment that the product uses when it is started and running.

S

sample. The data that the monitoring agent collects for the monitoring server instance. The interval is the time between data samplings.

sampled event. Sampled events happen when a situation becomes true. Situations sample data at regular intervals. When the situation becomes true, it opens an event, which gets closed automatically when the situation goes back to false (or when you close it manually). See also “event” on page 218.

Secure Sockets Layer. A security protocol for communication privacy that provides secure client/server conversations. SSL provides transport layer security (authenticity, integrity, and confidentiality) for a secure connection between a client and a server.

seed data. The product-provided situations, templates, policies, and other sample data included
with a monitoring agent to initialize the Tivoli Enterprise Monitoring Server's Enterprise Information Base. Before you can use a monitoring agent, the monitoring server to which it reports must be seeded, that is, initialized with application data.

server.  An application that satisfies data and service requests from clients.

SELinux.  The National Security Agency's security-enhanced Linux (SELinux) is a set of patches to the Linux kernel plus utilities that together incorporate a strong, flexible mandatory access control (MAC) architecture into the kernel’s major subsystems. SELinux enforces the separation of information based on confidentiality and integrity requirements, which allows attempts to tamper with or bypass application security mechanisms to be recorded and enables the confinement of damage caused by malicious or flawed applications.

Simple Network Management Protocol.  A TCP/IP transport protocol for exchanging network management data and controlling the monitoring of network nodes in a TCP/IP environment. The SNMP software protocol facilitates communications between different types of networks. IBM Tivoli Monitoring uses SNMP messaging to discover the devices on your network and their availability.

Simple Object Access Protocol.  The Simple Object Access Protocol is a lightweight, XML-based interface that vendors use to bridge remote procedure calls between competing systems. SOAP makes it unnecessary for sites to choose between CORBA/Java/EJB and Microsoft's COM+.

Because XML and SOAP are platform- and language-neutral, users can mix operating systems, programming languages, and object architectures yet maintain business-component interoperability across platforms: using SOAP, applications can converse with each other and exchange data over the Internet, regardless of the platforms on which they run.

single sign-on.  This feature lets your IBM Tivoli Monitoring users start other Tivoli web-enabled applications from any Tivoli Enterprise Portal client (desktop, browser, or Java Web start), or to start the Tivoli Enterprise Portal from those applications, without having to re-enter their user credentials (userid and password). For SSO to function, User Authentication must be configured through the Tivoli Enterprise Portal Server for an external LDAP registry (such as Microsoft Active Directory) that is shared by all participating Tivoli applications. All the participating applications must be configured for SSO and must belong to the same security domain and realm.

situation.  The set of monitored conditions running on a managed system that, when met, creates an event. A situation comprises an attribute, an operator such as greater than or equal to, and a value. It can be read as “If system_condition compared_to value is_true”. An example of a situation is: If CPU_usage > 90% TRUE. The expression “CPU_usage > 90%” is the situation condition.

Situation Event Forwarder.  This component of Tivoli Management Services maps situation events to Event Integration Facility (EIF) events and uses the EIF interface to send the events to a Tivoli Enterprise Console event server or to an OMNIbus EIF probe (the EIF receivers). The event receivers receive the forwarded events, and expand and format them for the event servers. On the Tivoli Enterprise Console or OMNIbus console, users can view, acknowledge, or reset situation events. The updated situation status is returned to the originating hub monitoring server and reflected in the Tivoli Enterprise Console or OMNIbus console.

Situation Update Forwarder.  The Situation Update Forwarder is a Java- and CORBA-based background process for communication between IBM Tivoli Enterprise Console and a particular Tivoli Enterprise Monitoring Server running under IBM Tivoli Monitoring version 6.1 and subsequent releases. Your site must install this component on the Tivoli Enterprise Console server; for instructions, see the IBM Tivoli Enterprise Console Installation Guide.

SNMP.  See “Simple Network Management Protocol.”

SOAP.  See “Simple Object Access Protocol.”

sockets.  Refers to the sockets method of passing data back and forth between a networked client and server or between program layers on the same computer.

sound.  The WAV file that plays whenever a situation becomes true for the current Navigator item. Sound is assigned to the Navigator item for a situation in the same way a state is assigned.

SPIPE.  A secure pipe is an implementation of the Internet Protocol's pipe specification that uses the Secure Sockets Layer API. Using SPIPE, your corporate Web server can securely access internal servers that are not based on the HTTPS protocol, while retaining their ability to process HTTP requests.

SQL.  See “Structured Query Language” on page 224.

SSL.  See “Secure Sockets Layer” on page 222.

SSM.  See “System Service Monitors” on page 224.

SSO.  See “single sign-on.”

state.  The severity of the situation event: critical, warning, or informational. Indicated by a colored event indicator, state is set in the Situation editor and can be different for different Navigator items.

status.  The true or false condition of a situation.
Structured Query Language. A standards-based programming language for extracting information from and updating information within a relational database. The Query editor enables you to write SQL queries to ODBC data sources for retrieval and display in table and chart views.

subnetwork. A configuration wherein a single IP network address is split up so it can be used on several interconnected local networks. Subnetting is a local configuration; outside it appears as a single IP network.

SUF. See “Situation Update Forwarder” on page 223.

Summarization and Pruning agent. One of the IBM Tivoli Monitoring base agents, the Summarization and Pruning agent keeps the data warehouse from growing too large by summarizing and pruning your stored historical data at intervals you set. For every attribute group that has data collection configured, you specify how often to aggregate (summarize) data in the Tivoli Data Warehouse and the length of time to delete (prune) data from the warehouse.

symbol. Represents a variable that can be added to header or footer text for data views, expert-advice text, or query specification. The detailed attribute name is enclosed in dollar signs, such as $SORIGINNODES$, and resolves to the attribute’s value. For Tivoli Enterprise Monitoring Server queries, == $NODE$ specifies the managed systems from which to retrieve data. For queries to be used in link target workspaces, you can create symbols for attributes using the $symbolname$ format.

System Monitor Agent. These agents were introduced with IBM Tivoli Monitoring V6.2.2 for nodes that run the desktop operating systems (Windows, Linux, UNIX). These agents operate only autonomously (that is, they run without a connection to a Tivoli Enterprise Monitoring Server) and pass SNMP trap data about operating system performance to an SNMP Event Collector such as IBM Tivoli Netcool/OMNIbus’s MTRAPRD receiver.

No other IBM Tivoli Monitoring agents or other components should be installed on the same node as a System Monitor Agent. The only exception to this rule is agents created with the Agent Builder tool for V6.2.2 or subsequent.

System Service Monitors. The IBM Tivoli Netcool/OMNIbus product provides System Service Monitors that support basic system-level monitoring of network components such as operating systems. Additionally, OMNIbus provides ISMs (Internet Service Monitors) and ASMs (Application Service Monitors).

Take Action. A Tivoli Enterprise Portal dialog box from which you can enter a command or choose from a list of predefined commands. It also lists systems on which to effect the command, which is usually a response to an event.

Take Action command. A Take Action command allows you to send commands to your managed systems, either automatically, in response to a situation that has fired (that is, turned true), or manually, as the Tivoli Enterprise Portal operator requires. With Take Action commands, you can enter a command or select one of the commands predefined by your product and run it on any system in your managed network. Thus you can issue Take Action commands either against the managed system where the situation fired or a different managed system in your network.

target libraries. SMP/E-controlled libraries that contain the data installed from the distribution media.

task. A unit of work representing one of the steps in a process.


TDW. See “Tivoli Data Warehouse.”

telnet. A terminal emulation program used on TCP/IP networks. You can start a telnet session with another system and enter commands that execute on that system. A valid userid and password for that remote system are required.

TEPS/e. See “Tivoli Enterprise Portal Server extended services” on page 225.

threshold. (1) A level set in the system at which a message is sent or an error-handling program is called. For example, in a user auxiliary storage pool, the user can set the threshold level in the system values, and the system notifies the system operator when that level is reached. (2) A customizable value for defining the acceptable tolerance limits (maximum, minimum, or reference limit) for an application resource or system resource. When the measured value of the resource is greater than the maximum value, less than the minimum value, or equal to the reference value, an exception is raised.

timestamp. A data type that contains a seven-part value that consists of a date and time expressed in years, months, days, hours, minutes, seconds, and microseconds.

Tivoli Data Warehouse. This member of the IBM Tivoli Monitoring product family stores Tivoli Monitoring agents’ monitoring data in separate relational database tables so you can analyze historical trends using that enterprise-wide data. Reports generated from Tivoli Data Warehouse data provide information about the availability and performance of your monitored environment over different periods of time.
**Enterprise Portal**

**Tivoli Integrated Portal.** The client component of a Tivoli Enterprise Portal Server. The Tivoli Enterprise Portal provides the graphical user interface into monitoring data collected by the Tivoli Enterprise Monitoring Server and prepared for user display by the portal server. The Tivoli Enterprise Portal comes in two versions: the desktop client and the browser client.

**Tivoli Enterprise Portal Server.** The server you log onto and connect to through the Tivoli Enterprise Portal client. The portal server connects to the hub Tivoli Enterprise Monitoring Server; it enables retrieval, manipulation, and analysis of data from your enterprise's monitoring agents.


**Tivoli Enterprise Web Services.** An open standards-based interface to the monitoring server that uses SOAP requests. Using SOAP, any monitoring agent can be dynamically queried, which means that its performance and availability data can be processed by external applications not a part of IBM Tivoli Monitoring.

**Tivoli Integrated Portal.** The Tivoli Integrated Portal provides a single, task-based navigation panel for Web-based products; it offers a common user interface with which you can launch applications, launch from one application to another, and share information between applications. It provides a rich, Web-based user interface with which you can access results generated by such applications as Tivoli Application Dependency Discovery Manager (TADDM), Tivoli Business Service Manager (TBSM), and Netcool/OMNibus. The Tivoli Integrated Portal enables the secure passing of data between Tivoli products by using this common portal. With it, you can view different aspects of your managed enterprise using a single dashboard view.

**Tivoli Management Services.** An integrated, layered architecture consisting of data-access, communication, and presentation components that enable cross-platform operation and integration of enterprise-wide data for systems-management applications. The software foundation that supports the development and operations of the Tivoli Enterprise Monitoring Server, the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal, and their monitoring agents.

**Transmission Control Protocol/Internet Protocol.** An open, portable communications protocol that is the software basis for the Internet.

**TSO.** Time Sharing Option, the interactive interface into the z/OS operating system.

**U**

**User Datagram Protocol.** A TCP/IP communications protocol that exchanges messages ("datagrams") between networked computers linked by the Internet Protocol (IP). UDP is an alternative to the Transmission Control Protocol (TCP), which, like UDP, uses IP to move a message from one computer to another. Unlike TCP, however, UDP does not divide the message into packets and reassemble them at the other end. The Network File System uses UDP to move file contents and file updates between the NFS server and the NFS client.

**UDP.** See [User Datagram Protocol](#).

**V**

**value of expression.** A function in a situation condition, query specification, or data view filter or threshold that uses the raw value of an attribute. A value can be a number, text string, attribute, or modified attribute. Use this function with any operator.

**view.** A window pane, or frame, in a workspace. It may contain data from an agent in a chart or table, or it may contain a terminal session or browser, for example. A view can be split into two separate, autonomous views.

**W**

**Warehouse Proxy agent.** One of the IBM Tivoli Monitoring base agents, the Warehouse Proxy agent passes historical monitoring data from either a monitoring agent or the Tivoli Enterprise Monitoring Server to the Tivoli Data Warehouse. This multithreaded server process can handle concurrent requests from multiple data sources to roll off data from their short-term history files to the data warehouse.

**WAV file.** Waveform audio format for storing sound in files, developed jointly by Microsoft and IBM.

**wildcard.** An asterisk (*) used to represent any characters that may follow or precede those entered, such as Sys* to find System and SysTray. Used in formulas with the VALUE function or MISSING function (in the Missing Task List). Used also with the SCAN function, but at the beginning of the text as in *Z to find markZ and typeZ.
Windows Management Instrumentation. Microsoft's Windows Management Instrumentation API provides a toolkit for managing devices and applications in a network of Windows-based computers. WMI provides both the data about the status of local or remote computer systems and the tools for controlling the data. WMI is included with the Windows XP and Windows Server 2003 operating systems.

WMI. See "Windows Management Instrumentation."

workload. A percentage that shows how much of its resources a managed system is using. Workload is calculated using a weighted combination of resource use statistics.

workspace. The viewing area of the Tivoli Enterprise Portal window, excluding the Navigator. Each workspace comprises one or more views. Every Navigator item has its own default workspace and may have multiple workspaces.

workspace administration mode. A global parameter set in the Administer Users editor but which is available only for userids with administrator authority. When enabled for a userid, customization of workspaces, links, and terminal-session scripts automatically becomes available to all users connected to the same Tivoli Enterprise Portal Server.

X

XML. See "Extensible Markup Language" on page 218.

Z

z/OS. IBM's operating system for its line of mainframe, zSeries computers known for its processing speed and its ability to manage large amounts of memory, direct-access storage, and data.
Index

Special characters
&bsz variable 190
&expdsn variable 190
&pdsn variable 190
&rhilev variable 190
&act variable 190
&ssz variable 190
&SYSCLONE. system variable 91
&SYSNAME system variable 90
&unit2 variable 190

Numerics
3270 Model 2 display requirements 173
3270 OMEGAMON interface 50

A
Add Runtime Environment panel
  base 86
  full 86, 89
  sharing 86, 89
ADDFILE command 198
address translation
  broker partitions 29
  enabling 29
  external address 29
  internal address 29
Advanced Encryption Standard 215
AES
  See Advanced Encryption Standard
agents
  See monitoring agents
alert monitor 215
Alignment field 193
APF-authorization
  enable 68
APF-authorizing the load libraries 105
Application field 192
application server 216
Application Service Monitors, Netcool/OMNIbus 224
application support 108
applids, VTAM
  model 91
  prefix 91, 92
  wildcard suffixes 91
archived historical data
  accessing 188
  connecting 189
  disconnecting 189
  name of persistent data store group 188
  restoring 188
audit logging
  overview 9
authentication
  3270 OMEGAMON interface 50
  CUA interface 50

authentication (continued)
  LDAP registry 50
  local registry 50
  menu interface 50
  OMEGAMON II interface 50
  passwords 50
  Service Console users 51
  SOAP server users 51
  users 50
authorization
  users 50
  well-known port 30
autonomous agents 23

B
BACKUP command 196, 198
BACKUP logic 196
baroc files 216
base libraries 35
base runtime environments 37, 39, 86
Basic Recorder of Objects in C files
  See baroc files
batch jobs
  See also CICATB job
  JCL 80
  log 82
  REGION parameter 80
  SCAN option 124, 156
  SCAND option 124
batch mode
  enabling 121
  job 122
  parameter format 180
  parameters 125
  scenarios 139, 148
  utilities 178
    KCICFKEY to manage PF keys 179
    KCICPGHP to display batch mode help 180
    KCISET for setting up the environment 178
BATCHLIB library 123, 150
BATCHMEM parameter member 123, 150
best practices 16, 22
broker partitions 29
browser client 216
building runtime libraries 95

C
CANCTDCS logmode 92
CBPDO 11
CISUPG2 member 83
CICATB job
  See also batch jobs
  creating 122, 149
  data dictionary 124, 130
  location 150

© Copyright IBM Corp. 2008, 2012
CICATB job (continued)

reports 124
running to validate changes 156
SCAN option 124, 156
SCAND option 124
SUBMIT options 123, 150

CIM
See Common Information Model

Classic linking feature
See Dynamic XE to 3270 (Classic) linking feature

Classic OMEGAMON interface
See 3270 OMEGAMON interface

client/server architecture 216

CLIST, executing in TKANCUS library 174
CMS_NODEID environment variable 20
codes, product 201
Col Row Len field 192
Column Count field 192
column description record 192
Column Length field 193
Column Map field 193
Column Name field 193

commands
ADDFILE 198
BACKUP 198
COMMINT 200
DELFILE 198
EXTRACT 196, 199
INITDS 199
ISPF RECOVERY OFF 173
persistent data store 197
PFSHOW 173
QUERY CONNECT 200
QUERY DATASTORE 200
RECOVER 199
RECOVERY OFF 173
RESUME 199
SWITCH 196, 197

COMMIT command 200
Common Event Console 6
Common Information Model 216
Common Object Request Broker Architecture 216, 223

communication protocols
decisions 27
limitations 28
piped 28
port numbers 31
SNA 27, 91
SOAP server 22
TCP/IP 27, 28, 31, 92

communications, securing 49
completing the configuration 101, 103
components
codes 201
overview 3
configuration
completing 101, 103
Configuration Tool 77
copying values 89
decisions 15
planning 15

configuration (continued)
runtime environments 37
security, runtime environment 91
setting up Configuration Tool 77
verifying 108

Configuration Assistance Tool
See Configuration Tool
configuration methods
Configuration Tool (ICAT) 25
PARMGEN Workflow user interface 25

configuration profile (PARMGEN) 63
configuration profile parameters
validate using PARMGEN 73

Configuration Tool
batch mode 121
batch utilities 178
DEBUG options 174
display requirements 78, 173
displaying an ISPF table 174
dynamically generated content 79edit recovery 79
environment 77, 78, 79
executing a CLIST in TKANCUS library 174
help 79
ISPF features 79
online help 79
panel IDs 79
PF keys 78
recovery 79
reference information 173
restrictions 173
rules 173
Runtime Environments (RTE) panel 83, 180
services 173
setup 77
split screen 79
starting 78

system variable support
creating the VTAM major node rename job 170
enabling 168
parameter member 169
sample usage 167
unlocking high-level qualifiers 173

usage notes 78
utilities 151, 173
work environment 79
work libraries 82

Configuration Tool (ICAT) configuration method 25
Configuration Tool environment
cloned 77
new 78

Configuration Tool panels
Add Runtime Environment 86, 89, 93
Configure Products 83, 168, 169, 170
Create/Clone Batch Mode Parameters 126, 152
Create/Clone Parameter Deck Menu 151
KCIMBATO 151
KCIMRTU 151
KCIPBATO 152
KCIPIN1 80
KCIPIN2 80

228 IBM Tivoli OMEGAMON XE and IBM Tivoli Management Services on z/OS: Common Planning and Configuration Guide
Configuration Tool panels (continued)
- KCIPPLST 84, 168, 169, 170
- KCIPPR4 82
- KCIPRCM 83, 168, 169, 170
- KCIPRDME 182
- KCIPRIM 78
- KCIPRTA1 89
- KCIPRTA2 93
- KCIPRTAB 86
- KCIPRTE 84
- KCIPRTEA 86
- Main Menu 78
- Product Selection Menu 84, 168, 169, 170
- README Table of Contents Display 182
- RTE Utility Menu 151
- Runtime Environments (RTEs) 84
- Set up Configuration Environment 82
- Set up Work Environment 80
- Specify JCL Options 80
- Specify Nonstandard Parameters panel 99
- Configuration Tool, z/OS 216
- configuration verification report 96
- Configure Products panel 83, 168, 169, 170
- console views 6
- CORBA
  See Common Object Request Broker Architecture
- COUNT option 32, 33
- Create/Clone Batch Mode Parameters panel 126, 152
- Create/Clone Parameter Deck Menu panel 151
- CSI
  cloned 77
  decisions 15
  existing 77
  new 78
  shared 15
- CTDDSN node ID 104
- CUA interface 50
- customization options 63
- Customize PARMGEN configuration profile members panel 64

D
DASD
See storage
data dictionary, CICATB 124, 130
Data Encryption Standard 217
Data Length field 193
Data Offset field 193
data record format, exported historical data 190
data sets
- INSTDATA 82
- INSTDATW 82
- INSTJOBS 82, 83
- INSTLIB 82
- INSTLIBW 82
- INSTQLCK 83
- INSTSTAT 83
- naming convention 35
- SYSTCPD 103
- VTAMLST 104
- data store
  See persistent data store
- Data Store Path field 191
- Data Store Ver field 191
- data, historical
  See historical data collection
database server 216
ddname KBDXTRA 218
debugging 174
decisions
  communication protocols 27
  configuration methods 25
  CSI 15
  high-availability hub 18
  historical data collection 45
  monitoring agents 22
  placement of monitoring servers 16
  platform for hub monitoring server 16
  runtime environments 35
  security 48
  user interface 23
default certificate 50
- DELFILE command 198
- DELFILE statement 189
deployment
decisions 15
  planning 15
- DES
  See Data Encryption Standard
desktop client 217
developerWorks website 210
diagnostic SYSOUT class 89
dictionary header record 191
Dictionary Len field 192
dictionary mappings, historical data 191
disk space
  See storage
- Display PARMGEN environment analysis panel 61
display requirements 173
- DLL
  See Dynamic Link Library
documentation
  See publications
- DVIPA 18, 28
  DWL feature
  See Dynamic XE to 3270 (Classic) linking feature
Dynamic Link Library 217
dynamic VIPA
  See DVIPA
Dynamic Workspace Linking (DWL)
  See Dynamic XE to 3270 (Classic) linking feature
Dynamic XE to 3270 (Classic) linking feature 94

E
EBCDIC files, exporting to 193
edit recovery 79, 173
EGG1 encryption scheme 50
EIB
  See Enterprise Information Base
EIF

See Event Integration Facility
End-to-End Response Time collector 6
enhancements 8
Enterprise Information Base 218
time situation 218
environment variables
See also parameters
CMS_NODEID 20
KDE_TRANSPORT 32, 33
KDEB_INTERFACELIST 33
ephemeral pipe support (EPS)
defined 29
disabling 29
forcing ephemeral connections 29
errors, parameter 156
Event Integration Facility 218, 221
event synchronization 6
Exclude products from PARMGEN customization
panel 61
Expansion Area field 191
Expansion field 192
Export field 191
Export Version field 191
exporting historical data 190, 191
Extensible Markup Language 218
EXTRACT command 196, 199
EXTRACT logic 196
extracting historical data to flat files 193

F

features, new 8
FIPS 197 215
firewalls
address translation 28
basic implementation 28
broker partitions 29
ephemeral pipe support (EPS) 28
gateway support 30
piped protocols 28
with IPPIPE protocol 28
with SOAP 20
Flag field 193
flat file, extracting historical data to 193
FMIDs 13
folding passwords to uppercase 91
format, product tapes 11
fragmentation, VSAM control area 16
full runtime environments 37, 38, 86, 89
function keys 173

gateway, firewall 30
georeferenced map 218
global SAF class name 91
Global Security Toolkit
See GSKit
Global Security Toolkit (GSKit) 49
global table, SOAP server 21

H

Header Len field 192
header record 195
help, Configuration Tool 79
HFS/zFS USS directory
update 69
hidden parameters 98
Hierarchical File System/zSeries File System
(HFS/zFS) 94
high-availability hub 9, 18
high-level qualifiers
length 35
non-VSAM libraries 87
shared CSI 16
unlocking 173
VSAM libraries 87
historical data
accessing 188
exporting 190
restoring 188, 190
historical data collection
allocating space 46
decisions 45
defined 45
enable 117
interval 45
limiting amount 45
location 46
managing 47
planning 45
host name
runtime environment 92
Telnet listener 94
Hostname field 92, 94
hot-standby monitoring server 218
HTTP 219
HTTPS 219
HTTPS protocol 34, 49, 218, 223
hub monitoring server
decisions 16
defined 4
high-availability 18
location 16
platform 16
SOAP server list 21
Hypertext Transport Protocol
See HTTP
Hypertext Transport Protocol over SSL (HTTPS)
See HTTPS protocol

I

IBM Support Assistant 211
IBM Tivoli Management Services on z/OS
See Tivoli Management Services
IBM Tivoli OMEGAMON enhanced 3270 user interface 4, 8, 24, 48
See OMEGAMON enhanced 3270 user interface
ICAT 216
See Configuration Tool
ICSF
See also Integrated Cryptographic Service Facility
See Integrated Cryptographic Service Facility (ICSF)
Identifier Num field 192
IEBGENR utility 189
IIOP
See Internet Inter-ORB Protocol (IIOP)
IIOP protocol
See Internet Inter-ORB Protocol (IIOP)
iKeyMan 50
INITDS command 196, 199
installation
checking for fix packs 11
media 11
Installation and Configuration Assistance Tool
See Configuration Tool
INSTDATA data set 78, 82
INSTDATW data set 78, 82
INSTJOBS data set 78, 82, 83
INSTJOBS library 150
INSTLIB data set 78, 82
INSTLIBW data set 78, 82
INSTLOG data set 78, 82
INSTQLCK data set 78, 83
INSTSTAT data set 78, 83
integral Web server 219
Integrated Cryptographic Service Facility 218
Integrated Cryptographic Service Facility (ICSF) 49
Integrated Service Management Library 210
integration with other products 7
interfaces
OMEGAMON (3270)
OMEGAMON II (CUA) 24
interfaces, network 33
interim libraries (PARMGEN) 63
Internet Inter-ORB Protocol 219
Internet Inter-ORB Protocol (IIOP) 49
Internet Service Monitors, Netcool/OMNIbus 224
interoperability 7
Interoperable Object Reference 219
Interoperable Object Reference (IOR) protocol 49
IOR
See Interoperable Object Reference (IOR) protocol
IOR protocol
See Interoperable Object Reference (IOR) protocol
IP address 92
IP.PIPE protocol
address translation 28
ephemeral pipe support (EPS) 28
with firewalls 28
ISA 211
ISPF
display requirements 173
displaying table 174
edit recovery 79
PANELID command 79
ISPF (continued)
RECOVERY command 79
RECOVERY OFF command 173
split screen 79
IVP
See configuration verification report

J
Java Database Connectivity 219
Java Management Extensions API 219
JCL
options 80
REGION 80
stored 83
suffix 36, 88
JDBC
See Java Database Connectivity
JMX
See Java Management Extensions API
job index (PARMGEN) 62
jobs, naming convention 36

K
KCICFKEY utility 179
KCICPGHP utility 180
KCICjALO job 74
KCICjCOPY job 75
KCICjIVP job 75
KCICjLNK job 75
KCICjLOD job 74
KCICjSEC job 74
KCICjSYS job 75
KCICjUPV job 75
KCICjUSP job 75
KCICjUSS job 75
KCICjW2R job 75
KCIMBATO panel 151
KCIMRTU panel 151
KCIP@PG0 panel 57
KCIP@PG2 panel 59
KCIP@PG3 panel 59
KCIP@PG4 panel 61
KCIP@PG5 panel 61
KCIP@PG6 panel
customize 64
KCIPBATO panel 152
KCIPIN1 panel 80
KCIPIN2 panel 80
KCIPPLST panel 84, 168, 169, 170
KCIPPR4 panel 82
KCIPRCM panel 83, 168, 169, 170
KCIPRDME panel 182
KCIPRIM panel 78
KCIPRTA1 panel 89
KCIPRTA2 panel 93
KCIPRTAB panel 86
KCIPRTE panel 84, 180
KCIPRTEA panel 86
KCISESETUP utility 178
load libraries
  APF-authorizing 105
  RKANMOD 105
  RKANMODL 105
  RKANMODP 105
  RKANMODU 105
  TKANMOD 105
  TKANMODL 105
  TKANMODP 105
load optimization 88
log, configuration batch jobs 82
logmode table 92
low-level qualifiers 35
LPAR-specific libraries 35
LU6.2 logmodes 92
LUGROUP field 94

M
Main Menu panel 78
managed Java beans
  See MBeans
Map Length field 192
maps, parameter 96
MBeans 219, 220
members, user-modified 89
menu interface 50
menu, main 78
Mgmtclas field 81, 87
Microsoft Management Console 220
mid-level qualifiers 35, 88
migration
  See upgrade
mixed-case passwords 91
MMC
  See Microsoft Management Console
model applids, VTAM 91
monitoring agents
  address space 22
  address spaces 22
  autonomous 23
  base 6
  completing the configuration 103
  configure using PARMGEN 70
declared 6
OMEGAMON 6
overview 4
packaging 13
stand-alone 22
monitoring server
  See Tivoli Enterprise Monitoring Server

N
naming
  backup historical data sets 188
data sets, runtime environment 35
jobs, runtime environment 36
members, JCL 36
TEMS name 20

languages
  packs 13
  support 13, 108
LDAP
  See Lightweight Directory Access Protocol
LDAP registry 50
Length field 191
libraries
  base 35
  Configuration Tool 82
  IBM Tivoli Monitoring 208
  INSTJOBS 150
  LPAR-specific 35
  OMEGAMON XE shared publications 207
runtime
  building 95
  defined 35
  loading 100
SMP/E target 35
target 35
TKANCUS 174
types 35
work 82
Lightweight Directory Access Protocol 219
listening port numbers
  See port numbers
lists
  hub monitoring servers 21
  user IDs 21
NAT
See Network Address Translation
national languages
See languages
navigation 53
Netcool/OMNibus 224
MTTRAPD trap receiver for 224
Netcool/OMNibus ObjectServer 6
NETID parameter 92
NetView
Take Action command authorization 10
Network Address Translation 220
Network File System 220
network ID 92
network interface list 33
new features 8
Newest Row field 192
NFS
See Network File System
NOFF option 193
nonstandard parameters 98

O
ODBC
See Open Database Connectivity
Oldest Row field 192
OMEGAMON (3270) interface 24
OMEGAMON enhanced 3270 user interface
configure using PARMGEN 70
considerations 48
enable security 112
features 4, 8
requirements 24
OMEGAMON II (CUA) interface 24
OMEGAMON II interface 50
OMEGAMON subsystem 6
OMEGAMON Tivoli Event Adapter 221
OMEGAMON XE
components 3
overview 3
OMEGAMON XE monitoring agents
packaging (continued)
tapes 11
Tivoli Management Services 11
PANELID command 79
panels, Configuration Tool
See Configuration Tool panels
parameter map 96
parameters
See also environment variables
batch mode 125
changed 157
changing 96, 98
CMS_NODEID 20
CSI_DSN 59
data dictionary 124, 130
effects 156
GBL_INST_HILEV 59
GBL_SYSDA_UNIT 59
GBL_TARGET_HILEV 59
GBL_USER_JCL 56
hidden 98
Install Job Generator (JOBGEN) output library 58
JOBCARD data 58
KDE_TRANSPORT 32, 33
KI2.11nn_CLASSIC_IMSID 166
KI2.11nn_CLASSIC_MPFREFIX 166
maps 96
modifying 96, 98
nonstandard 98
OPT(xxx) 64
REGION 80
REPLACE 63
reports 96
RTE profile library and RTE member name 58
RTE_HILEV 60, 166
RTE_NAME 57, 165
RTE_PLIB_HILEV 57
RTE_SHARE 166
RTE_SMS_MGMTCLAS 60
RTE_SMS_PDSE_FLAG 60
RTE_SMS_STORCLAS 60
RTE_SMS_UNIT 60
RTE_SMS_VOLUME 60
RTE_SMS_VSAM_MGMTCLAS 60
RTE_SMS_VSAM_VOLUME 60
RTE_VSAM_HILEV 60, 166
setting 96, 98
system variable parameter member 169
TARGET_ZONE 59

PARMGEN
complete set up 73
configuration profile 63
configuring a Tivoli Enterprise Monitoring Server 70
configuring an OMEGAMON enhanced 3270 user interface 70
configuring OMEGAMON monitoring agents 70
conversion job 63
create runtime environment members and jobs 73
customize configuration profiles 64
enable APF-authorization statements (Optional) 68
enable self-describing agent feature 69

Index 233
PARMGEN (continued)
interim libraries 63
job index review 62
prerequisites 55
starting the PARMGEN Workflow user interface 56
submit batch jobs 73
update application definitions 67
update global parameters 71
update global VTAM major node 67
update SMS values 69
update started tasks names 66
update TCP/IP port values 67
update the HFS/zFS USS directory 69
update user prolog (optional) 65
update VTAM node 67
validate profile parameter values 73
verify products 65
PARMGEN configuration method
 overview 8
PARMGEN configuration profiles
 customize 64
PARMGEN Workflow user interface
 set up system variables 165
 starting 56
PARMGEN Workflow user interface configuration
 method 25
partitions 29
passwords
 authentication, configuring 50
 mixed-case 91
 uppercase 91
PDPROC prefix 185
PDS
 See persistent data store
PDSE field 81, 88
PDSLOG SYSOUT 190
PerfMon
 See Performance Monitor, Windows
Performance Monitor, Windows 221
performance, effect of fragmentation 16
persistent data store 218
 access to files 116
 archived data group name 188
 archived data, accessing 188
 automatic maintenance 186
backup data sets, naming 188
command interface 197
commands 197
components
 KPDARCH 195
 KPDDSCO 196
 KPDMANE 195
 KPDMREST 195
 KPDUUTIL 195
 KPDXTRA 196
connecting archived data sets 189
controlling maintenance 187
data record format 190
data set backup 187
disconnecting archived data sets 189
export data sets, naming 187
persistent data store (continued)
extracting data to EBCDIC files 194
extracting data to flat files 193
maintaining 47
overview 185
restoring data 190
user ID for running procedures 186
Persistent Datastore 221
PF keys 173
PFSHOW command 173
piped protocols
 firewall support 28
limitations 28
platforms 16
PMAP
 See parameter map
POOL option 32, 33
Port number field 93, 94
port numbers
 algorithm 31
 allocating 31
 assigning 31
 changing 32
 COUNT option 32
default 31
POOL option 32
range 33
reserving 31, 32
SKIP option 32
Telnet listener 94
portal client
 See Tivoli Enterprise Portal client
portal server
 See Tivoli Enterprise Portal Server
ports
 opaque 33
 well-known 30, 31, 32, 93
prefix
 started task procedures 88
 VTAM applid 91, 92
prerequisites
 Configuration Tool 173
display 173
security 48
TCP/IP protocols 28
preventive service planning (PSP) maintenance 85
private situation 222
probes, shared 7
product codes 201
product integration 7
product registration
 automatic 9
Product Selection Menu panel 84, 168, 169, 170
protocols
 See communication protocols
publications
 developerWorks website 210
hardcopy 11
Integrated Service Management Library 210
publications (continued)
  OMEGAMON XE 207
  Redbooks 210
related 209
shared 207
  Technotes 210
  Tivoli Management Services on z/OS 207
types 207
wikis 210

Q
qualifiers, high-level
  See high-level qualifiers
QUERY CONNECT command 200
QUERY DATASTORE command 200

R
RACF
  MIXEDCASE option 91
  passwords 91
README Table of Contents Display panel 182
RecID field 191, 192, 193
RECOVER command 199
RECOVER logic 196
RECOVERY command 79
RECOVERY OFF command 173
Redbooks 210
REGION parameter 80
remote monitoring server
  configuring in batch mode 148
  decisions 16, 19
  defined 4
  location 16, 19
  reporting to hub on z/OS 148
Remote Procedure Call 222
Remote RTE for Transport field 88
reports
  CICATB job 124
  configuration verification 96
  parameter maps 96
requirements
  See also prerequisites
communication protocols 22, 27
Configuration Tool 173
display 173
DVIPA 28
firewall support 28
security 48
SOAP server 20
TEMS name 20
Tivoli Management Services 11
VIPA 28
RERUN option 123, 150
RESUME command 199
RESUME logic 197
RKANSAMU data set 103
Row Count field 192
Row Length field 193
RPC
  See Remote Procedure Call
RTE
  See runtime environments
RTE directory field 94
RTE Utility Menu panel 151
runtime environment members and jobs
  create using PARMGEN 73
runtime environments
  actions required 182
  Add option 180
  adding 85
  base 37, 39, 86
  Build option 180
  building libraries 95
  completing the configuration 103
  Configuration Tool 83, 180
  Configuration Tool options 180
  Configure option 180
  configuring 37
  convert 63
  copying configuration values 89
  creating 85
data sets 35
decisions 35
defined 35
defining 85
defining using shared DASD 133
deleting 183
developing 35
existing 85
full 37, 38, 86, 89
JCL 36
jobs 36
libraries 35
load optimization 88
Load option 180
name, system variable 90
naming convention
  data sets 35
  jobs 36
new 85
optimization 88
options 180
overview 35
replicating 139
security 91
self-contained 37, 38
shared DASD 133
sharing 86, 89
sharing with base 37, 40, 90
sharing with full 37, 42
sharing with SMP/E 37, 43
transporting 88, 131, 134
  batch mode parameters 136
  from one image to another 133
types 36
Update option 180
update TCP/IP port values 67
Utilities option 180
verify products 65
View option 180
runtime environments (continued)
  VTAM major node 170
Runtime Environments (RTEs) panel 84, 180
runtime libraries
  building 95
  defined 35
  loading 100
runtime members, analyzing 89

S
  sample procedures 103
  SCAN option 123, 124, 150, 156
  SCAND option 123, 124, 130, 150
  scenarios
    batch mode 139, 148
    remote monitoring server with z/OS hub 148
    replication of configured environment 139
  Secure Hypertext Transport Protocol
    See HTTPS protocol
  secure Hypertext Transport Protocol (HTTPS)
    See HTTPS protocol
  secure pipe
    See SPIPE
  secure protocols
    HTTPS protocol 34, 49
    Internet Inter-ORB Protocol (IIOP) 49
    Interoperable Object Reference (IOR) protocol 49
  Secure Socket Layers (SSL) 34, 49, 218, 219, 223
  Secure Sockets Layer API 222
  security
    3270 OMEGAMON interface 50
    command-level 50
    communications 49
    CUA interface 50
    enable 111
    firewall implementation 28
    menu interface 50
    OMEGAMON enhanced 3270 user interface 112
    OMEGAMON II interface 50
    prerequisite tasks 48
    product-level 50
    requirements 48
    runtime environment 91
    Service Console 51
    SOAP server 21, 51
    UNIX System Services (USS) 94
  security-enhanced Linux
    See SELinux
  Security: Validate User option 50
  self-describing agent
    enable 69
    feature overview 8, 17, 23
  SELinux 223
  sequential file, exporting persistent data to 190
  ServerPac 11
  Service Console user authentication 51
  Set up Configuration Environment panel 82
  Set up PARMGEN work environment for a runtime environment panel 57, 59

SET UP PARMGEN WORK ENVIRONMENT FOR AN RTE panel 59
Set up Work Environment panel 80
shared CSI
  advantages 15
  guidelines 16
shared probes 7
shared publications 207
sharing runtime environments
  system variables 90
  with base 37, 40
  with full 37, 42
  with SMP/E 37, 43
short-term historical data
  See persistent data store
Simple Network Management Protocol 216, 219, 221, 223
Simple Object Access Protocol 223
Simple Object Access Protocol (SOAP) 20
  See SOAP server
single sign-on 50, 223
Situation Event Console 6
Situation Update Forwarder 221, 223
SKIP option 32, 33
Slot Size field 191
SMP/E
  See also System Modification Program/Extended
    sharing target libraries 37, 43
    target libraries 35
SMS considerations 81, 87
SNA protocol
  requirements 27
  VTAM communication values 91
SNMP
  See Simple Network Management Protocol
SOAP
  See Simple Object Access Protocol
SOAP server
  aliasing 21
  communication protocols 22
  configuration 21
  defined 20
  global table 21
  hub monitoring server list 21
  in firewall environments 20
  requirements 20
  security 21, 51
  terminology 21
  user access 21
  user access list 21
  web services 9
Software Support 211
  space requirements, historical data collection 46
  Spare 1 field 193
  Spare field 193
  Specify JCL Options panel 80
  Specify Nonstandard Parameters panel 99
  specifying
    global SAF class name 91
  SPIPE 218, 223
  SPIPE protocol 49
Tivoli Enterprise Monitoring Server (continued)
high-availability hub 18
hub
defined 4
high-availability 18
location 16
name 20
placement of hub 16
placement of remote 16, 19
platform decision 16
remote
configuring in batch mode 148
defined 4
location 16, 19
reporting to hub on z/OS 148
TEMS name 20
verifying the configuration 108
VIPA name 28
Tivoli Enterprise Portal 225
authorizing users 50
client, defined 5
enable historical data collection 117
Tivoli Enterprise Portal Server 5
Tivoli Enterprise Portal Server extended services
See TEPS/e
Tivoli Enterprise Portal Server extended services
(TEPS/e) 6
Tivoli Integrated Portal 225
Tivoli Management Services
components 3
distributed components 3
overview 3
packaging 11
required software 11
TKANCUS library 174
TLS
See Transport Layer Security API
TMS:Engine, defined 6
TN3270 Telnet sessions 93
tool, configuration
See Configuration Tool
Total Slots field 191
Transmission Control Protocol/Internet Protocol 225
Transport Layer Security API 219
transporting runtime environments
batch jobs 134
batch mode parameters 136
between z/OS images 133
methods 131

U
UDP
See User Datagram Protocol
Unit field 81, 87
UNIX System Services (USS) 94
unlocking high-level qualifiers 173
uppercase passwords 91
Used Slots field 191
user access list 21
User Datagram Protocol 225
user IDs 50
persistent data store procedures 186
SOAP server access 21
sysadmin 48
validation 21
user interfaces
3270 “green screen” 23
Tivoli Enterprise Portal interface
OMEGAMON enhanced 3270 user interface 23
user-modified members 89
users
authenticating 50
authorizing 50
utilities, Configuration Tool 151

V
validation
See authentication
Variable Cols field 192
variables, environment
See environment variables
variables, system
See system variables
verifying the configuration 108
VIPA
dynamic 28
monitoring server name 28
requirements 28
Virtual IP Addressing
See VIPA
Volser field 81, 87
VSAM files
control area splits 16
fragmentation 16
managing 16
performance considerations 16
VSAM libraries
high-level qualifiers 87
VTAM
applids
model 91
prefix 91, 92
wildcard suffixes 91
communication values 91
definitions 104
logmode table 92
major node
rename job 170
runtime environment 170
varying active 105
network ID 92
VTAMLST data set 104
VTAMLST startup member 92

W
Warehouse Proxy agent 5
WAV file 225
WCONFIG(KCJJPUP1) job 63
Web Services
  See SOAP server
well-known port
  authorization  30
  changing  32
  default number  31, 93
wikis  210
wildcard  225
wildcard suffixes, VTAM applids  91
Windows Dynamic Link Library
  See Dynamic Link Library
Windows Management Instrumentation API  226
Windows Performance Monitor
  See Performance Monitor, Windows
WMI
  See Windows Management Instrumentation API
work environment  79
work libraries  82
work libraries (PARMLIB)  57
Workload Manager (WLM)
  started task settings  204

X

XML
  See Extensible Markup Language