

IBM Tivoli Event Pump for z/OS
Version 4.2.2

Installation and Configuration Guide



IBM Tivoli Event Pump for z/OS
Version 4.2.2

Installation and Configuration Guide



Note

Before using this information and the product it supports, read the information in “Notices” on page 203.

This edition applies to Version 4.2.2 of IBM Tivoli Event Pump for z/OS (product number 5698-B34) and to all subsequent versions, releases, and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2011.**

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

Contents

Figures	xi
--------------------------	-----------

About this publication	xiii
Intended audience	xiii
Publications	xiii
IBM Tivoli Event Pump for z/OS library	xiii
Related publications	xiii
Accessing terminology online	xiv
Accessing publications online	xiv
Ordering publications.	xiv
Accessibility	xiv
Tivoli technical training	xv
Tivoli user groups	xv
Support information	xv
Conventions used in this publication	xv
Typeface conventions	xv
Operating system-dependent variables and paths	xvi
Syntax diagrams	xvi

Part 1. Introduction	1
---------------------------------------	----------

Chapter 1. Introduction	3
--	----------

Event Pump overview	3
Event Pump components	4
Master address space	4
Source collector	5
Event distributor	5
External data interface	6
Event processing	6
SOAP interface	6
Data space address space	7
Event Pump data sources and event feeds	7

Chapter 2. New or changed functions for IBM Tivoli Event Pump for z/OS Version 4.2.2	9
---	----------

Chapter 3. Planning for installation	11
---	-----------

Data sources and event feeds	11
Software requirements	11
Configuration requirements	11
Security requirements	12
Access to HFS files	12
TCP/IP considerations.	12
SOAP requests	12
DB2 considerations	12
IMS considerations	12

Part 2. Installing Event Pump components	15
---	-----------

Chapter 4. Installation overview	17
---	-----------

Program Directory	17
Preventive Service Planning (PSP)	17
Applying a required PTF	17
Installation process overview	17

Installation results	18
Chapter 5. Preparing the MVS system	19
Using SMP/E to load the Event Pump libraries	19
Updating SYS1.PARMLIB members	19
MPFLSTxx (IMS only)	19
PROGxx	19
Assigning dispatching priority	20
Updating SAF	20
Chapter 6. Preparing the Event Pump program	21
Chapter 7. Installing and configuring the EIF probe	23
Installing the EIF probe (Tivoli Netcool/OMNIBus 7.2.1 or earlier)	23
Installing the EIF probe (Tivoli Netcool/OMNIBus 7.3.0 or later)	24
Copying files from the installation package	25
Installing the schema updates	25
Configuring the Netcool/OMNIBus ObjectServer	26
Installing and configuring the EIF probe rules	27
Rule file include structure	27
Installation steps.	28
For Windows systems (Netcool/OMNIBus server files)	28
For UNIX and Linux systems (Netcool/OMNIBus server files)	29
Configuration steps.	29
Chapter 8. Getting ready to start the Event Pump program	31
Updating the GPZPUMP startup procedure	32
Updating the EIF configuration file	36
Updating the GPZDSPC startup procedure	36
Chapter 9. Verifying the Installation	39
Verify that EIF logging is enabled	39
Starting and stopping Event Pump procedures	39
Starting and stopping the data space	39
Starting and stopping the master address space	40
Review the AOPEDLOG log.	40
Part 3. Enabling data sources and event feeds	41
Chapter 10. Data source and event feed summary	45
Chapter 11. Data sources	47
JOBLOG	47
Specifying JES SPOOL data sets that you want to monitor.	47
Adding traps for the JOBLOG data source	49
Registering JES JOBLOG as a message source	49
JES3 environment restrictions	49
SYSLOG	50
Specifying SYSLOG messages that you want to monitor	50
Registering the SYSLOG as a message source	51
Chapter 12. Automation	53
Tivoli AF/OPERATOR.	53
Installing the AF/OPERATOR EDI interface	53
Reviewing the Event Pump GPZPARMS member.	54
Customizing AF/OPERATOR to send events	54
Calling the EDI	54
Starting and ending the automation process	54

Additional modification examples	55
Error handling	56
Modifying the source	56
Tivoli NetView for z/OS automation	57
Tivoli System Automation for z/OS	58
Event flow.	59
Installing System Automation for z/OS	59
Updating the System Automation for z/OS control file	59
Customizing the NetView program to enable Event Pump to process events.	60
Specifying the resources to be monitored	60
Customizing System Automation for z/OS to send events.	60
Calling the EDI	61
Starting and ending the automation process	61
Additional examples of requests	62
Error handling	62
Modifying the source	63
Restarting the NetView program	64
Verifying the installation	64
Disable logging the events that are sent to the PPI receiver	65
BMC MainView AutoOPERATOR for z/OS.	65
Installing the AutoOPERATOR EDI interface	65
Reviewing the Event Pump GPZPARMS member.	66
Customizing BMC MainView AutoOPERATOR to send events	66
Calling the EDI	66
Starting and ending the automation process	66
Additional modification examples	67
Error handling	68
Modifying the source	68
Computer Associates OPS/MVS	69
Installing the CA OPS/MVS EDI interface	70
Reviewing the Event Pump GPZPARMS member.	70
Customizing the CA OPS/MVS event feed to send events.	71
Calling the EDI	71
Starting and ending the automation process	71
Additional examples of requests	72
Error handling	73
Modifying the source	73
Chapter 13. BMC MainView	75
BMC MainView for z/OS.	75
Event flow.	75
Event scenario	76
Configuring BMC MainView for z/OS	76
Installing Event Pump for BMC MainView z/OS components	76
Registering BMC MainView for z/OS	76
Enabling event processing	76
Disabling event processing	76
BMC MainView for CICS.	77
Event flow.	77
Event scenarios	78
DB2 CICS interface unavailable.	78
DB2 CICS interface becomes available again	78
Configuring BMC MainView for CICS	79
Registering BMC MainView for CICS.	79
Enabling event processing	79
Disabling event processing	79
BMC MainView for DB2	79
Event flow.	80
Event scenarios	81
TSO user exceeds its processor resource usage threshold	81
TSO user processor resource use drops beneath threshold	81

Configuring BMC MainView for DB2.	81
Registering BMC MainView for DB2	82
Enabling event processing	82
Disabling event processing	82
Chapter 14. CICS	83
Installing the CICS Transient Data Queue Exit program	83
Review the CICS started task	84
Reviewing the Event Pump GPZPARMS member.	84
Registering base CICS resources and CICS Transient Data	84
Chapter 15. CICSplex System Manager	85
Reviewing resource definitions	85
Defining System Availability Monitoring Alerts	86
Defining MAS Resource Monitoring Alerts..	86
Excluding resources and events.	87
Reviewing the GPZPUMP procedure	88
Registering CICSplex System Management source	88
Chapter 16. DB2 Universal Database for z/OS	91
Installing Event Pump for DB2 components	91
Adding DB2 sysplex support	91
Review security	92
Configuring each DB2 subsystem or data sharing group	92
Reviewing the GPZPUMP procedure	93
Registering the DB2 event feed for monitoring.	93
Enabling DB2 event processing	93
Disabling DB2 event processing	93
Excluding resources and events.	94
Supported resource monitoring.	94
Reasons for excluding resources or events	94
How to exclude resources or events	95
Hierarchy of exclude processing	95
Refreshing exclude rules	96
Show a list of Exclude Rules.	96
Syntax of the Exclude Rules	96
Chapter 17. Health Checker for z/OS	101
Registering Health Checker data feed for monitoring	101
Assigning Health Checker filtering and severities	101
Chapter 18. IMS	103
Installing the Event Pump for IMS components	104
Enabling the OTMA callable interface	104
Configuring the IMS security parameters	105
Installing the IHS\$MPF exit	105
Installing the IMS AO exit	106
Verify the AO exit installation	106
Determine coexistence issues with CA OPS/MVS	107
Configuring the program-to-program interface (PPI)	107
Configuring the NetView components	107
Verifying the installation	108
Verifying host customization	108
Verifying end-to-end connectivity.	108
Verifying IMS event monitoring	108
Verifying poll monitors	109
Disabling the logging of all Event Pump events to the NetView log	109
Customizing event processing	109
Customizing polling processing operation and thresholds	110
Excluding resources and events	112

Tuning for IMS monitoring	113
Chapter 19. Resource Management Facility (RMF).	115
Configuring RMF	115
Updating the master address space parameters (GPZPARMS)	115
Updating the registration data file (REGDATA)	115
Including RMF statements	116
Customizing RMF exception definitions	116
Chapter 20. Storage Management Resource Monitoring (SMS)	121
Registering SMS as a message source	121
Registering disk device and storage volume resources	122
Chapter 21. Tivoli NetView for z/OS	123
Installing Event Pump components	123
Determine the NetView for z/OS started tasks	123
Customize each NetView for z/OS started task	124
Updating NetView for z/OS started task JCL.	124
Configuring NetView parameter members.	124
Modifying security settings for the NetView program	126
Defining the program-to-program interface connection	127
Customize event feeds	128
Chapter 22. Tivoli Workload Scheduler for z/OS	129
Update Event Pump to receive events from Tivoli Workload Scheduler	129
Update Tivoli Workload Scheduler	129
Part 4. Configuring Event Pump	131
Chapter 23. Configuring registration data (REGDATA)	133
Update the registration data file	133
Registration data file example	134
Chapter 24. Filtering Event Pump for z/OS events.	137
Review the GPZPARMS member	137
Message selection and filtering within the Event Pump	137
Event record format	138
Default EIF select table	139
Using the EIF Event logging facilities	139
Starting and stopping the EIF component	140
Chapter 25. Configuring event-driven registration.	141
Review and update the GPZPUMP startup procedure	141
Update the EDR select table	142
EDR select table example	142
Distribute EDR traps to all Event Pump programs in a sysplex	142
Chapter 26. Creating user-defined events	143
Identifying the application to monitor	143
Identifying the messages to monitor.	144
Developing the message monitoring logic	145
Developing a handler	146
Predefined variables	147
Predefined fields	147
Creating a message mask member	149
Chapter 27. Updating message event severity and state using Netcool/OMNIBus . . .	151
Starting the Netcool/OMNIBus Administrator	151

Working with basic SQL commands	151
Selecting rows in the alerts.zos.severity.msgstate_lookup table	151
Adding new rows to the alerts.zos.severity.msgstate_lookup table	152
Deleting rows from the alerts.zos.severity.msgstate_lookup table	152
Updating rows in the alerts.zos.severity.msgstate_lookup table	152
Using the command-line interface	152

Part 5. Appendixes 153

Appendix A. Startup parameters (GPZPARMS) 155

Appendix B. Event Pump samples reference 159

Appendix C. Configuring the startup procedures using three address spaces 163

Updating the GTMPUMP startup procedure	164
Updating the GTMSVR startup procedure	166
Update the EIF configuration file	168
Updating the GTMDSPC startup procedure	168
Starting and stopping three address spaces	169
Starting and stopping the data space	169
Starting and stopping the event distributor	170
Starting and stopping the source collector	170

Appendix D. External Data Interface overview 173

EDI exit format	174
Tracing CALL statements	176
API exit routine return codes	176
Format types, action types, and data types	177
BMC MainView AutoOPERATOR for z/OS (format type 12)	177
CA OPS/MVS (format type 09)	178
Tivoli AF/OPERATOR (format type 10)	180
Tivoli System Automation for z/OS (format type 07)	181

Appendix E. Event Pump IMS AO exits 185

IHSIAOE0	186
IHSIAOEC controller exit	186

Appendix F. Migrating from TBSM Version 3 object pump to Event Pump Version 4

Release 2 189

Summary of changes for IBM Tivoli Event Pump for z/OS Version 4.2.1	189
Sysplex support	190
Registration data migration.	190
Enabling registration data capture	190
Converting the captured registration data	190
Adding group entries	191
Activating the new registration data.	191
Migration scenario: migrating registration data from TBSM Version 3 to Event Pump V4.2	191
Startup considerations for TBSM Version 3 and Event Pump Version 4.2 (dual mode)	192
Event Pump source collector startup parameters for TBSM Version 3 and Event Pump Version 4	192
Event Pump data space startup parameters for TBSM Version 3 and Event Pump Version 4	193
Event Pump event distributor startup parameters for TBSM Version 3 and Event Pump Version 4	194
Optional DD names for the TBSM event distributor for TBSM Version 3 and Event Pump Version 4	195
Setting parameters for language support for TBSM Version 3 and Event Pump Version 4	195
Filtering Event Pump for z/OS events for TBSM Version 3 and Event Pump Version 4	196
Event format	196
Message selection and filtering within the event distributor	196
Event logging	197
Implementation and configuration on the z/OS host	198
Updating the Event Pump event distributor JCL.	198

Updating the EIF configuration file	198
Updating the Event Pump event distributor startup parameters	199
The event distributor operator commands for the EIF interface for TBSM Version 3 and Event Pump Version 4	199
Supported data sources and event feeds	200
Notices	203
Trademarks	204
Index	207

Figures

1. The Event Pump for z/OS program	4	16. Source collector and event distributor address	
2. old.AlertGroup != 'TBSMV3_SOURCE390'	27	space data files	164
3. Rules file include structure	28	17. AF/OPERATOR EDI exit	174
4. Master address space data files	32	18. EDI exit CALL format	174
5. AOPEDLOG log example	40	19. Example Tivoli System Automation for z/OS	
6. Sample GTMMON member SYSLOG01	51	EDI API exit call	175
7. Registration data file example	134	20. Tracing Tivoli System Automation for z/OS	
8. USERMSGs member example	134	EDI API invocation statements	176
9. JOBLOG member example	135	21. Tracing Tivoli System Automation for z/OS	
10. Event Pump event record	138	specific EDI API invocation statements	176
11. OMNIBus event.	139	22. IHSIAOE0 Exit	185
12. EIF select table	139	23. IMS AO Exit Interface.	185
13. Sample EIF Log Output	140	24. Sample EIF Log Output	
14. Resulting Netcool/OMNIBus exception event	146	(EIF_LOG_LEVEL=BOTH)	198
15. Resulting Netcool/OMNIBus resolution event	146		

About this publication

IBM® Tivoli® Event Pump for z/OS® is an enterprise management product that monitors the z/OS data processing resources in an enterprise. You can use it to monitor the z/OS systems, subsystems, applications, and other resources that are critical to your business. This publication, the *IBM Tivoli Event Pump for z/OS Installation and Configuration Guide*, describes how to install and configure Event Pump for z/OS.

Intended audience

This publication is for system programmers, system administrators, network operators, and others who need installation and configuration information for Event Pump.

Publications

This section lists publications in the IBM Tivoli Event Pump for z/OS library and related documents. It also describes how to access Tivoli publications online and how to order Tivoli publications.

IBM Tivoli Event Pump for z/OS library

The following documents are in the IBM Tivoli Event Pump for z/OS library:

- *Command Reference*, SC27-4040, describes the commands for Event Pump for z/OS.
- *Configuration Reference*, SC27-4041, describes the configuration statements for Event Pump for z/OS.
- *Event Reference*, SC27-2895, lists the events (exceptions and messages) that are integrated with Event Pump for z/OS.
- *Installation and Configuration Guide*, SC27-4042, describes how to install and configure Event Pump for z/OS.
- *Message Reference*, GC27-2310, describes the messages for Event Pump for z/OS.
- *Troubleshooting Guide*, GC27-4043, provides information for use in diagnosing and solving problems that occur in Event Pump for z/OS.
- *Program Directory for IBM Tivoli Event Pump for z/OS*, GI11-8920, contains information about the material and procedures that are associated with installing Event Pump for z/OS.

Related publications

For information about the IBM Tivoli Business Service Manager product, link to the product information center from <http://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Tivoli+Business+Service+Manager>.

For information about the IBM Tivoli Netcool/OMNIBus product, link to the product information center from <http://www.ibm.com/developerworks/wikis/display/tivolidoccentral/OMNIBus>.

Accessing terminology online

The IBM Terminology website consolidates the terminology from IBM product libraries in one convenient location. You can access the Terminology website at <http://www.ibm.com/software/globalization/terminology>.

The following terms are used in this library:

AOPxxLOG

For the AOPxxLOG system log in the 2-address-space environment or the AOPLOG system log in the 3-address-space environment. AOPxxLOG is either AOPELOG (event distributor log) or AOPSCLOG (source collector log).

AOPLOG

For the AOPxxLOG system log in the 2-address-space environment or the AOPLOG system log in the 3-address-space environment. AOPxxLOG is either AOPELOG (event distributor log) or AOPSCLOG (source collector log).

Accessing publications online

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Documentation Central website at <http://www.ibm.com/tivoli/documentation>.

Note: If you print PDF documents on other than letter-sized paper, set the option in the **File → Print** window that allows Adobe Reader to print letter-sized pages on your local paper.

Ordering publications

You can order many Tivoli publications online at <http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, contact your software account representative to order Tivoli publications. To locate the telephone number of your local representative, perform the following steps:

1. Go to <http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>.
2. Select your country from the list and click **Go**.
3. Click **About this site** in the main panel to see an information page that includes the telephone number of your local representative.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. Event Pump for z/OS has no user interface or user interactions except through the interfaces of other products. For information about accessibility features for those products, see the applicable product information.

Visit the IBM Accessibility Center at <http://www.ibm.com/alphaworks/topics/accessibility/> for more information about IBM's commitment to accessibility.

Tivoli technical training

For Tivoli technical training information, refer to the following IBM Tivoli Education website at <http://www.ibm.com/software/tivoli/education>.

Tivoli user groups

Tivoli user groups are independent, user-run membership organizations that provide Tivoli users with information to assist them in the implementation of Tivoli Software solutions. Through these groups, members can share information and learn from the knowledge and experience of other Tivoli users.

Access the Tivoli Users Group at <http://www.tivoli-ug.org>.

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 the support you need:

Online

Access the Tivoli Software Support site at <http://www.ibm.com/software/sysmgmt/products/support/index.html?ibmprd=tivman>. Access the IBM Support Portal at <http://www.ibm.com/support/entry/portal/software>.

IBM Support Assistant

IBM Support Assistant is a free local software serviceability workbench that you can use to resolve questions and problems with IBM software products. Support Assistant provides quick access to support-related information and serviceability tools for problem determination. To install the Support Assistant software, go to <http://www.ibm.com/software/support/isa/>.

After you install Support Assistant, you can install the IBM Tivoli Event Pump for z/OS add-on. In Support Assistant, click **Update > Find new**. In the Find New Product Add-ons window, click **Tivoli** and then find and install the IBM Tivoli Event Pump for z/OS add-on.

Troubleshooting Guide

For more information about resolving problems, see the *Event Pump for z/OS Troubleshooting Guide*.

Conventions used in this publication

This publication uses several conventions for special terms and actions, operating system-dependent commands and paths, and command syntax.

Typeface conventions

This publication uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as **Tip:**, and **Operating system considerations:**)
- Keywords and parameters in text

Italic

- Citations (examples: titles of publications, diskettes, and CDs)
- Words defined in text (example: a nonswitched line is called a *point-to-point line*)
- Emphasis of words and letters (words as words example: "Use the word *that* to introduce a restrictive clause."; letters as letters example: "The LUN address must start with the letter *L*.")
- New terms in text (except in a definition list): a *view* is a frame in a workspace that contains data.
- Variables and values you must provide.

Monospace

- Examples and code examples
- File names, directory names, and path names
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This publication uses the UNIX convention for specifying environment variables and for directory notation.

When using the Windows command line, replace *\$variable* with *%variable%* for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. The names of environment variables are not always the same in the Windows and UNIX environments. For example, *%TEMP%* in Windows environments is equivalent to *\$TMPDIR* in UNIX environments.

Note: If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Syntax diagrams

Read syntax diagrams from left-to-right, top-to-bottom, following the horizontal line (the main path). This section describes how syntax elements are shown in syntax diagrams.

Symbols

The following symbols are used in syntax diagrams:

- ▶▶ Marks the beginning of the command syntax.
- ▶ Indicates that the command syntax is continued.
- | Marks the beginning and end of a fragment or part of the command syntax.
- ◀◀ Marks the end of the command syntax.

Parameters

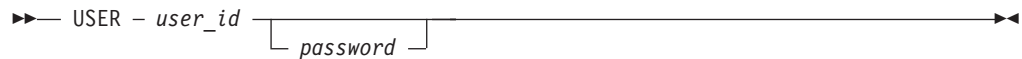
The following types of parameters are used in syntax diagrams:

- Required** Required parameters are shown on the main path.
- Optional** Optional parameters are shown below the main path.
- Default** Default parameters are shown above the main path. In parameter descriptions, default parameters are underlined.

Syntax diagrams do not rely on highlighting, brackets, or braces. In syntax diagrams, the position of the elements relative to the main syntax line indicates whether an element is required, optional, or the default value.

Parameters are classified as keywords or variables. Keywords are shown in uppercase letters. Variables, which represent names or values that you supply, are shown in lowercase letters.

In the following example, the `USER` command is a keyword, the `user_id` parameter is a required variable, and the `password` parameter is an optional variable.



Punctuation and parentheses

You must include all punctuation that is shown in the syntax diagram, such as colons, semicolons, commas, minus signs, and both single and double quotation marks.

When an operand can have more than one value, the values typically are enclosed in parentheses and separated by commas. For a single value, the parentheses typically can be omitted. For more information, see “Multiple operands or values” on page xviii.

If a command requires positional commas to separate keywords and variables, the commas are shown before the keywords or variables.

When examples of commands are shown, commas are also used to indicate the absence of a positional operand. For example, the second comma indicates that an optional operand is not being used:

```
COMMAND_NAME opt_variable_1,,opt_variable_3
```

You do not need to specify the trailing positional commas. Trailing positional and non-positional commas either are ignored or cause a command to be rejected. Restrictions for each command state whether trailing commas cause the command to be rejected.

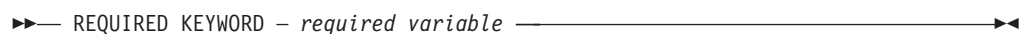
Abbreviations

Command and keyword abbreviations are listed in synonym tables after each command description.

Syntax examples

This section shows examples for the different uses of syntax elements.

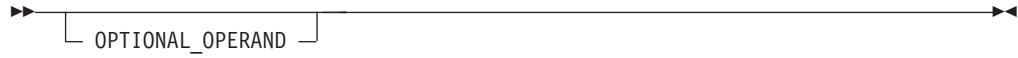
Required syntax elements: Required keywords and variables are shown on the main syntax line. You must code required keywords and variables.



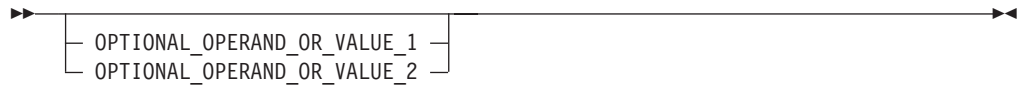
A required choice (two or more items) is shown in a vertical stack on the main path. The items are shown in alphanumeric order.



Optional syntax elements: Optional keywords and variables are shown below the main syntax line. You can choose not to code optional keywords and variables.

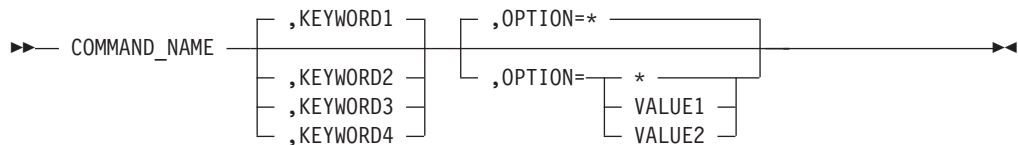


A required choice (two or more items) is shown in a vertical stack below the main path. The items are shown in alphanumeric order.

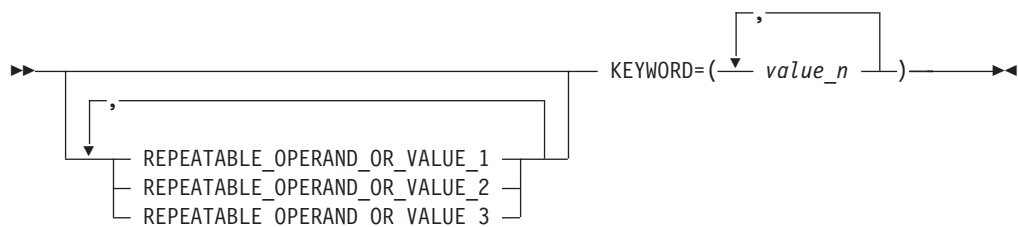


Default keywords and values: Default keywords and values are shown above the main syntax line in one of the following ways:

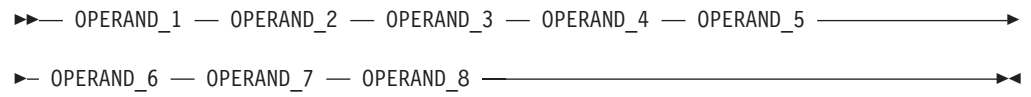
- A default keyword is shown only above the main syntax line. You can specify this keyword or allow it to default. The following syntax example shows the default keyword KEYWORD1 above the main syntax line and the rest of the optional keywords below the main syntax line.
- If an operand has a default value, the operand is shown both above and below the main syntax line. A value below the main syntax line indicates that if you specify the operand, you must also specify either the default value or another value shown. If you do not specify the operand, the default value above the main syntax line is used. The following syntax example shows the default values for operand OPTION=* above and below the main syntax line.



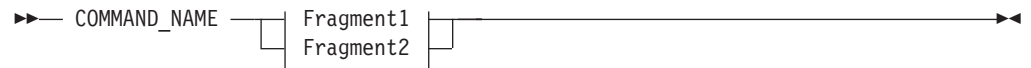
Multiple operands or values: An arrow returning to the left above a group of operands or values indicates that more than one can be selected or that a single one can be repeated.



Syntax that is longer than one line: If a diagram is longer than one line, each line that is to be continued ends with a single arrowhead and the following line begins with a single arrowhead.



Syntax fragments: Some syntax diagrams contain syntax fragments, which are used for lengthy, complex, or repeated sections of syntax. Syntax fragments follow the main diagram. Each syntax fragment name is mixed case and is shown in the main diagram and in the heading of the fragment. The following syntax example shows a syntax diagram with two fragments that are identified as Fragment1 and Fragment2.



Fragment1



Fragment2



Part 1. Introduction

Chapter 1. Introduction	3
Event Pump overview	3
Event Pump components	4
Master address space	4
Source collector	5
Event distributor	5
External data interface	6
Event processing	6
SOAP interface	6
Data space address space	7
Event Pump data sources and event feeds	7
 Chapter 2. New or changed functions for IBM Tivoli Event Pump for z/OS Version 4.2.2	 9
 Chapter 3. Planning for installation.	 11
Data sources and event feeds	11
Software requirements	11
Configuration requirements	11
Security requirements	12
Access to HFS files	12
TCP/IP considerations	12
SOAP requests	12
DB2 considerations	12
IMS considerations	12

Chapter 1. Introduction

IBM Tivoli Event Pump for z/OS can be used to monitor the z/OS systems, subsystems, applications, and other resources that are critical to your enterprise. For more information, see the following topics:

- “Event Pump overview”
- “Event Pump components” on page 4
- “Event Pump data sources and event feeds” on page 7

Event Pump overview

IBM Tivoli Event Pump for z/OS helps to provide an end-to-end view of your business environment by collecting the state of z/OS systems and subsystems. Event Pump processes messages from various sources and converts them to Event Integration Facility (EIF) events.

Event Pump collects z/OS state events from z/OS systems and subsystems (including CICS®, IMS™, and DB2® subsystems) and from the products of independent software vendors. Event Pump uses the Netcool/OMNIbus Probe for Tivoli EIF (the EIF probe) to forward these state events to IBM Tivoli Netcool/OMNIbus, where Netcool/OMNIbus alerts can be used by applications such as IBM Tivoli Business Service Manager (TBSM).

These EIF events can be used for event management in IBM Tivoli Netcool/OMNIbus, for business service management in TBSM, and for service management for z/OS systems in Integrated Service Management for System z®. TBSM also uses the information that is provided by Event Pump, coupled with resource discovery information from the z/OS discovery library adapter (DLA), to show changes in resources that are being monitored.

Event Pump is shown in Figure 1 on page 4. In this figure, DB2 indicates DB2 Universal Database for z/OS.

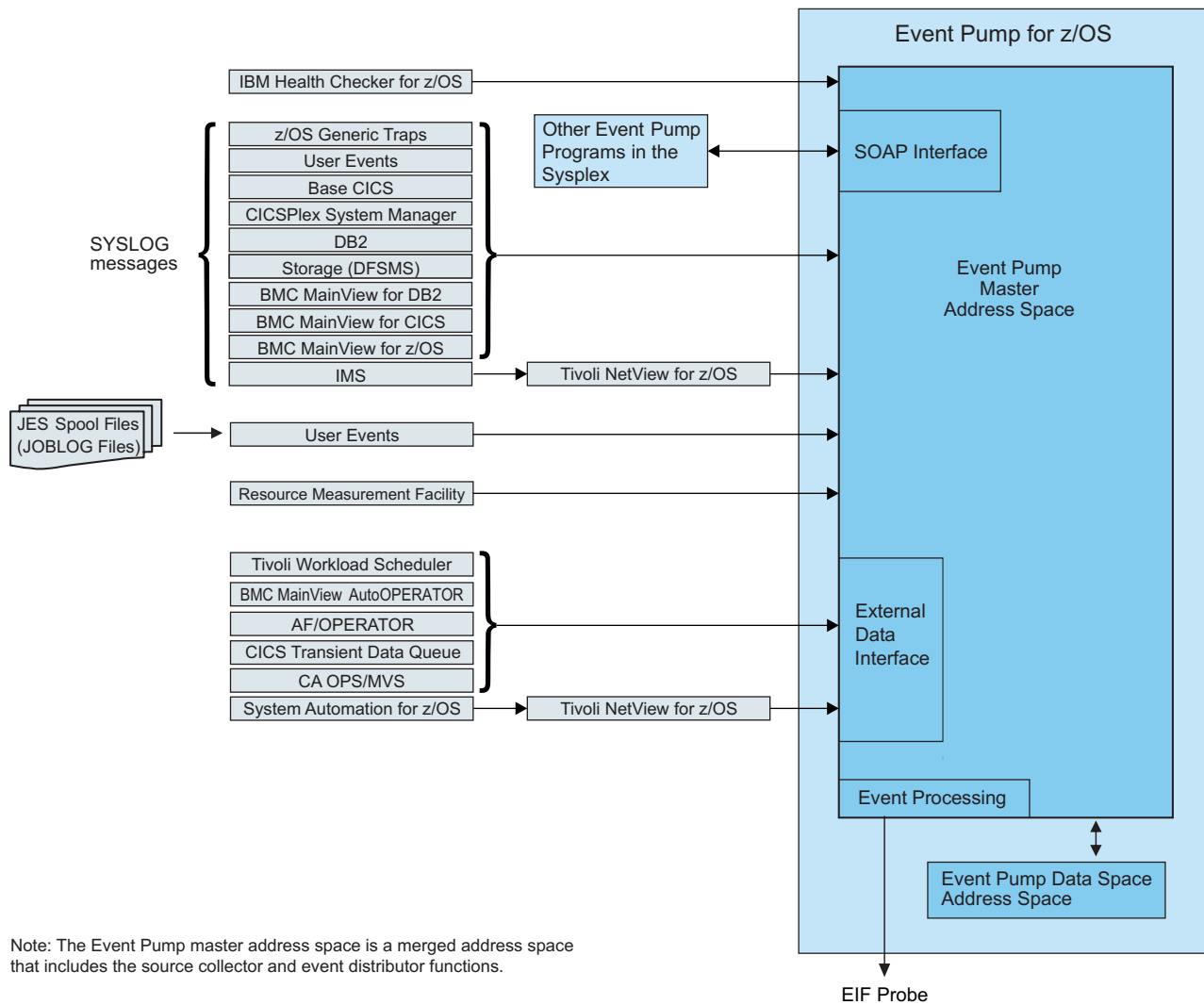


Figure 1. The Event Pump for z/OS program

Event Pump components

Event Pump for z/OS has the following address spaces. These address spaces must be running on each host operating system that you want to monitor.

- “Master address space”
- “Data space address space” on page 7

Master address space

The following components and functions run in the master address space:

- “Source collector” on page 5
- “Event distributor” on page 5
- “External data interface” on page 6
- “Event processing” on page 6
- “SOAP interface” on page 6

Source collector

The source collector captures and processes the following event data:

- The MVS™ console messages (SYSLOG messages)
- JES job log information (JOBLOG files)
- External data interface (EDI) information (for example, automation products)
- NetView® program-to-program interface (PPI) data
- Resource Measurement Facility™ and Health Checker for z/OS exceptions and information

The event data that is captured is based on the startup parameters of the master address space and on the resource registration data. The event data is placed in internal queues within the data space address space. The source collector formats this data, places the data in the data space, and notifies the event distributor that the data is available.

When an event is detected, the source collector formats a message according to a defined specification and queues the formatted message in the data space for the event distributor.

Information is made available to the source collector in the following ways:

- An extended MVS console is used to collect information for the MVS console (start, stop, or abend). The information that is collected includes information about started tasks, batch jobs, CICS regions, and DB2 Universal Database for z/OS.
- The external data interface (EDI) is used to integrate the Tivoli AF/OPERATOR, Tivoli System Automation for z/OS, Tivoli Workload Scheduler, CICS transient data queue, BMC MainView AutoOPERATOR for z/OS, and Computer Associates OPS/MVS products. State changes, messages, and exceptions (for resources) that reside on the same MVS operating systems are passed through the EDI to Event Pump. You can also send automation starting and automation ending messages to indicate that automation is in progress and that automation has completed for resources that are managed by Event Pump.
- The DB2 component of Event Pump works with DB2 management products. These messages and exceptions are passed directly to the source collector.
- The IMS management component of Event Pump that works with IMS management products and System Automation for z/OS use the NetView for z/OS program-to-program interface (PPI). These messages and exceptions are passed directly to the source collector.
- The Tivoli NetView for z/OS program also provides events to Event Pump. Events are captured by the NetView automation table and are sent to the source collector. The source collector adds the time, date, and sequence information to the preformatted record and moves it to the data space for transport to the event distributor.

Event distributor

The event distributor manages communication between the z/OS system on which Event Pump is running and the EIF probe. It retrieves event data from the data space and sends it to the EIF probe. It also passes the data that is gathered by the source collector to the EIF probe and logs any activity. The event distributor can filter events for transmission to the EIF probe.

The event distributor uses either LU 6.2 APPC services or TCP/IP services for this communication. The protocol that is used is set from initialization parameters when the event distributor starts.

External data interface

The external data interface (EDI) enables applications to send events to Event Pump that are related to the resources that these applications manage. The EDI uses cross-memory services to pass messages or exceptions that reside on the same z/OS operating system. The EDI inserts a time stamp and sequence number in a message or exception, adjusts the final length, and sends the message or exception to the source collector. The EDI provides an interface between the following products and Event Pump:

- Tivoli AF/OPERATOR on z/OS
- Tivoli System Automation for z/OS (with Tivoli NetView for z/OS)
- Tivoli Workload Scheduler for z/OS
- CICS transient data queue
- BMC MainView AutoOPERATOR for z/OS
- CA OPS/MVS

Event processing

Event processing involves capturing specific events and routing them to the EIF probe. The events are processed by the EIF probe and forwarded to the Netcool/OMNIbus ObjectServer server. Events can also trigger the detection of resources.

The EIF probe receives inbound z/OS events, formats them into Netcool/OMNIbus alerts according to probe rules, and forwards them to the Netcool/OMNIbus ObjectServer server as Netcool/OMNIbus alerts. The probe rules for z/OS event processing are organized by event feed (for example, System Automation for z/OS, DB2, CICS) using an include file structure. The probe rules can include a user override section, so that you can meet the needs of your particular enterprise.

The Netcool/OMNIbus ObjectServer server database receives and saves the Netcool/OMNIbus alerts. The Netcool/OMNIbus alerts.status table is updated with several columns for z/OS specific data. Triggers and lookup tables are used to assign severities and states to events, and to process message and resolution (clearing) events. The states are critical (red), warning (yellow), and normal (green). The severities are critical, high, medium, low, ignore, or unknown.

Physical resources that are detected by Event Pump can have two event types: messages and exceptions. Messages reflect the availability status of the resource (for example, active or inactive). Exceptions typically indicate that a resource is experiencing a temporary condition.

SOAP interface

The SOAP interface enables Event Pump to discover and communicate with other Event Pump for z/OS programs that are running in other LPARs (logical partitions) within the sysplex. Thus, Event Pump can ensure that registration for events occurs across the sysplex. The SOAP interface enables the following functions:

- Automatic discovery of Event Pump programs by each other when running in a sysplex environment
- Automatic propagation of event-driven registration (EDR) data to all other Event Pump programs within a sysplex environment, such that the restart of a failed job is detected wherever it runs in the sysplex
- Automatic resynchronization of EDR data within Event Pump at Event Pump startup
- DB2 data sharing support

Data space address space

The data space address space provides a temporary storage queuing area for passing data and information between the source collector and event distributor functions.

Event Pump data sources and event feeds

You can monitor data sources and event feeds. For a summary and the specifications of the data sources and event feeds, see Chapter 10, “Data source and event feed summary,” on page 45.

You can monitor the following data sources:

JOBLOG data source

You can specify JES spool files that you want to monitor and use the customizer to define user events, which are the specific messages that you want to trap.

SYSLOG data source

You can identify SYSLOG messages to monitor. You can also use the customizer to define user events (specific messages that you want to trap).

You can monitor the following event feeds:

BMC MainView

You can monitor the following BMC MainView products:

- BMC MainView for z/OS. You can monitor z/OS system events and messages.
- BMC MainView for CICS. You can monitor exceptions.
- BMC MainView for DB2. You can monitor exceptions.

CICS You can monitor base CICS events.

CICSplex® System Manager

You can monitor CICSplex System Manager events.

DB2 Universal Database™ for z/OS

You can monitor DB2 events.

EDI event feeds

You can monitor the following products using the EDI interface:

- CICS transient data queue. You can monitor exceptions.
- Tivoli Workload Scheduler. You can monitor messages and exceptions.
- Tivoli AF/OPERATOR. You can monitor text-based messages.
- Tivoli System Automation for z/OS. You can monitor events by type (batch, CICS, CMAS, DB2, IDMS, IMS, Tivoli OMEGAMON®, and started task).
- BMC MainView AutoOPERATOR for z/OS. You can monitor text-based messages.
- CA OPS/MVS. You can monitor text-based messages.

Health Checker for z/OS

You can monitor exceptions that are issued when a check detects a potential problem or a deviation from a suggested setting. You can also monitor information that is issued to the message buffer to indicate either a clean check run (no exceptions found) or an inappropriate check for the current environment that cannot run.

IMS You can monitor IMS default exceptions.

NetView for z/OS

You can monitor messages or exceptions that are passed from console-written NetView for z/OS REXX programs.

Resource Measurement Facility

You can monitor z/OS system events and messages

Storage Management Resources (SMS)

You can manage events for disk devices, storage groups, and storage volumes. You can also perform interval monitoring of exceptions.

Chapter 2. New or changed functions for IBM Tivoli Event Pump for z/OS Version 4.2.2

These new or changed functions are available with IBM Tivoli Event Pump for z/OS Version 4.2.2:

- New event feeds:
 - IBM Tivoli AF/OPERATOR on z/OS
This event feed was made available with the Version 4.2.1 interim feature.
 - Health Checker for z/OS.
This event feed was made available with the Version 4.2.2 feature pack.
 - BMC MainView:
 - MainView for z/OS
 - MainView for CICS
 - MainView for DB2
 - MainView AutoOPERATORThese event feeds were made available with the Version 4.2.1 interim feature.
 - Storage Management Subsystem (SMS) resource monitoring that provides SMS message and exception monitoring.
- New data source (JOBLOG).
JOBLOG monitoring extends message management by using application JES spool log files. Event Pump supports messages from the system console. It also supports messages in the JES output data sets to which every job writes during application processing. Major applications that write to application logs include important state and status information that can help in displaying the state and status of these applications in IBM Tivoli Netcool/OMNIbus or IBM Tivoli Business Service Manager (TBSM).
- Several customizer enhancements, including support for a %INCLUDE statement in the monitor member and support for user-supplied variables.
- Enhanced logging that improves problem analysis in JOBLOG data source monitoring and the customizer.
- Merging the source collector and event distributor address spaces to make Event Pump implementation faster and simpler by shortening the path length and eliminating cross address space communication.

The library was reorganized and contains the following books:

- *Event Pump for z/OS Installation and Configuration Guide*
- *Event Pump for z/OS Command Reference*
- *Event Pump for z/OS Configuration Reference*
- *Event Pump for z/OS Event Reference*
- *Event Pump for z/OS Message Reference*
- *Event Pump for z/OS Troubleshooting Guide*
- *Program Directory for IBM Tivoli Event Pump for z/OS*

A compressed file that contains the books in PDF format was added.

For IBM Tivoli Event Pump for z/OS Version 4.2.1 enhancements, see “Summary of changes for IBM Tivoli Event Pump for z/OS Version 4.2.1” on page 189.

Chapter 3. Planning for installation

Before installing and configuring the Event Pump product, make sure that you verify and identify the following information:

- Verify the data source and event feed levels of software that you plan to use.
- Identify the information that you require to configure the started tasks and samples.
- Identify the security requirements.

Data sources and event feeds

List the data sources and event feeds that you are including in your implementation. Include the resource types and the mapping to events.

For the z/OS environment, define the method for receiving events. Multiple data sources and event feeds can lead to confusion. You can have multiple events for the same problem. For example, for started tasks, information can be received using Event Pump, as well as from System Automation for z/OS. In this case, automation most likely generates the most critical information. Because of this, set the automation events to a priority higher than the base console messages for starting and stopping.

After you decide which resources to monitor, you can use the following methods to select those resources:

Auto detection

Some Event Pump resources are detected automatically. Currently supported examples include IMS and DB2 resources.

Detecting by event

This process detects resources based on events (messages and exceptions) that are sent to Event Pump. If a resource is not detected by auto detection and an event is received for this resource, the resource is registered and the received event is posted against the resource.

Resources can also be detected as a result of messages and exceptions that are issued by monitored products, for example IMS and DB2.

Software requirements

Before installing and configuring the Event Pump product, make sure the you have the correct levels of software for the event feeds. For information on the program requirements, see the *Program Directory for IBM Tivoli Event Pump for z/OS*.

Configuration requirements

Decide on the high-level qualifier names to use for the data sets that are used in the Event Pump Job Control Language (JCL) samples:

- Tivoli Event Pump data sets
- REXX alternate runtime library
- CPSM authorized library
- TCP/IP libraries

To configure the EIF probe, identify the following information:

- Port number for the Event Integration Facility (EIF) probe.
- Host name of the server on which the EIF probe is running

If you use the SOAP server to send requests to another Event Pump program in the same sysplex, identify the following information:

- SMF ID
- Port number for the Event Integration Facility (EIF) probe.
The default port number is 5530.
- Host name of the server on which the EIF probe is running

Security requirements

Before you begin installation, consider these security requirements:

- “Access to HFS files”
- “TCP/IP considerations”
- “SOAP requests”
- “DB2 considerations”
- “IMS considerations”

Access to HFS files

Before you use the EIF facility to send events to the IBM Tivoli Netcool/OMNIBus probe, authorize Event Pump to access HFS files. For more information on authorizing the SAF user ID, see “Updating SAF” on page 20.

TCP/IP considerations

Event Pump must be able to use the host z/OS system TCP/IP stack. The SAF user ID that is used by the master address space (or the event distributor address space) requires an OMVS segment and authorization to access TCPIP services on each LPAR on which the Event Pump is to run.

SOAP requests

If you plan to validate SOAP requests (using the `VALIDATE_SOAP_USERS` start up parameter), you must implement the RACF[®] `PASSTICKET` function. For more information on implementing the `PASSTICKET` function, see “Updating SAF” on page 20.

DB2 considerations

DB2 has the following security considerations:

- The DSNR resource class controls access to DB2 subsystems. If this class is enabled, give READ access to the *db2id*.BATCH profiles that represent DB2 subsystems. Event Pump requires this access to monitor the protected DB2 subsystems.
- Make sure that the Event Pump user ID has the authority to issue DB2 DISPLAY commands, including the DISPLAY DB and DISPLAY GROUP commands.
- If DB2 SQL discovery is required, make sure that the Event Pump user ID has the authority to read SYSIBM tables.

IMS considerations

IMS has the following security considerations:

- Define the following NetView operators to your security product: IHSIAUT0, IHSIAUT1, IHSIAUT2, and IHSIAUT3.
- Define the IHSAUT01 user ID to your security product. The IHSAUT01 user ID requires access to TCP/IP FTP.
- Define the OTMA Command Interface to your security product.
- To submit batch jobs, provide the appropriate access to the user ID under which the NetView region runs, and to the IHSAUT01 user ID.

For additional security information, see “Configuring the IMS security parameters” on page 105.

Part 2. Installing Event Pump components

Chapter 4. Installation overview	17
Program Directory	17
Preventive Service Planning (PSP)	17
Applying a required PTF	17
Installation process overview	17
Installation results	18
Chapter 5. Preparing the MVS system	19
Using SMP/E to load the Event Pump libraries	19
Updating SYS1.PARMLIB members	19
MPFLSTxx (IMS only)	19
PROGxx	19
Assigning dispatching priority	20
Updating SAF	20
Chapter 6. Preparing the Event Pump program	21
Chapter 7. Installing and configuring the EIF probe	23
Installing the EIF probe (Tivoli Netcool/OMNIbus 7.2.1 or earlier)	23
Installing the EIF probe (Tivoli Netcool/OMNIbus 7.3.0 or later)	24
Copying files from the installation package	25
Installing the schema updates	25
Configuring the Netcool/OMNIbus ObjectServer	26
Installing and configuring the EIF probe rules	27
Rule file include structure	27
Installation steps	28
For Windows systems (Netcool/OMNIbus server files)	28
For UNIX and Linux systems (Netcool/OMNIbus server files)	29
Configuration steps	29
Chapter 8. Getting ready to start the Event Pump program	31
Updating the GPZPUMP startup procedure	32
Updating the EIF configuration file	36
Updating the GPZDSPC startup procedure	36
Chapter 9. Verifying the Installation	39
Verify that EIF logging is enabled	39
Starting and stopping Event Pump procedures	39
Starting and stopping the data space	39
Starting and stopping the master address space	40
Review the AOPEDLOG log	40

Chapter 4. Installation overview

Before starting the installation and configuration procedures, perform the SMP/E installation that is described in the Event Pump program directory.

Workstation-based Event Pump code is included on a DVD. For service updates to workstation-based Event Pump code, see the Tivoli Software Support site at <http://www.ibm.com/software/sysmgmt/products/support/index.html?ibmprd=tivman>.

Program Directory

The *Program Directory for IBM Tivoli Event Pump for z/OS* is provided with the product and is also available from the Tivoli Documentation Central website at <http://www.ibm.com/tivoli/documentation>. The program directory provides instructions to install the Event Pump product to the SMP/E libraries. The program directory also contains helpful hints for a problem-free installation. It lists the program temporary fixes (PTFs) that enable other products to work properly with Event Pump. *Read the Event Pump program directory to help ensure a problem-free installation.*

Preventive Service Planning (PSP)

If problems are found, they are collected by IBM Software Support in what is called the *PSP bucket*. When you receive your copy of the Event Pump program, call IBM Software Support to request a copy of the PSP bucket. For additional information, see the Event Pump program directory.

In the United States, the telephone number for IBM Software Support is 1-800-237-5511. If you have an IBMLINK subscription, you can receive a copy of the PSP bucket through that service.

Applying a required PTF

The Event Pump program directory lists all APARs from previous releases that were created for z/OS and other IBM Tivoli network and Systems Management products to work with this Event Pump release. It also lists APARs from previous Event Pump releases that are incorporated into this release.

There might be required PTFs. You can get these PTFs by using SMP/E Internet Service Retrieval or by calling IBM Software Support. For information about SMP/E Internet Service Retrieval, see the *z/OS Introduction and Release Guide*. In the United States, the telephone number for IBM Software Support is 1-800-237-5511.

Installation process overview

Installation and configuration of Event Pump includes the following steps:

1. Installing the Event Pump program from the distribution media using z/OS SMP/E using the Event Pump program directory
2. Customizing and running sample jobs to install the basic functions of the Event Pump program

3. Updating startup procedures
4. Configuring the startup parameter members
5. Configuring registration data
6. Filtering events
7. Configuring event-driven registration data
8. Defining additional event feeds
9. Installing and configuring the EIF probe

Migrating users: For migration considerations, see Appendix F, “Migrating from TBSM Version 3 object pump to Event Pump Version 4 Release 2,” on page 189.

Installation results

When you complete the installation procedures, the following functions are operational:

- Base Event Pump components
- EIF probe
- Data sources and event feeds

Chapter 5. Preparing the MVS system

Follow these steps to prepare your MVS system if you are installing the Event Pump program:

- Use SMP/E to load the Event Pump libraries.
- Update the SYS1.PARMLIB members and activate any changes that you made.
- Assign a dispatching priority to the address spaces.
- Update the system authorization facility (SAF).

Using SMP/E to load the Event Pump libraries

Use the information in the Event Pump program directory to load the Event Pump libraries from the distribution media.

Usage note: The DDDEF statements in the GTMJDDF job might need to be modified for your environment. Refer to the comments in this job for further information.

Updating SYS1.PARMLIB members

Review the following members in the SYS1.PARMLIB data set:

- “MPFLSTxx (IMS only)”
- “PROGxx”

MPFLSTxx (IMS only)

The Event Pump IMS event feed uses the MCS interface to issue commands to IMS DBCTL regions. Output from these commands is displayed in the system log (SYSLOG), which can flood the system log for large DBCTL regions. If you are using large DBCTL regions, you can install the sample IHS\$MPF exit to prevent this output from being displayed in the system log. See “Installing the IHS\$MPF exit” on page 105 for information about the IHS\$MPF exit sample.

PROGxx

The PROGxx member contains the names of program libraries that you want the system to concatenate to the SYS1.LINKLIB data set and libraries that you want to define as authorized with the Authorized Program Facility (APF). Use the PROGxx member instead of using both the LNKSTxx member (to concatenate program libraries to the SYS1.LINKLIB data set) and the IEAAPFxx member (to authorize program libraries to APF).

The following data sets must be APF-authorized:

- SGTMMODS
- SCEERUN (Language Environment® services used by the Event Pump interface for the Event Integrated Facility–EIF)

To be authorized, a library name and volume ID must be in the list of authorized libraries in the PROGxx member of the SYS1.PARMLIB data set.

If you specify an unauthorized library on a STEPLIB DD statement or concatenate unauthorized libraries with authorized libraries, all libraries are treated as if they are unauthorized.

To add the Event Pump libraries to the list of APF-authorized libraries without requiring an MVS system restart, follow these steps:

1. Enable dynamic APF changes by issuing the following MVS command:
`SETPROG APF,FORMAT=DYNAMIC`
2. Add each Event Pump library to the list of APF-authorized libraries. For each Event Pump library that you want to add, issue the following MVS command:
`SETPROG APF,ADD,DSNAME=dsname,VOLUME=volser`

If you want information about...	Refer to...
APF authorization	z/OS library

Assigning dispatching priority

The Event Pump address spaces are automatically marked non-swappable through the SYSEVENT macro within the product code. Do not mark the programs non-swappable through the z/OS Program Properties table. Give the address spaces a dispatching priority like an online CICS region. If Event Pump does not have adequate performance levels, communication problems can occur.

Updating SAF

The SAF user ID that is used by the master address space (or the event distributor address space) requires the following authorization:

- Contain an OMVS segment
- WRITE authorization for the cache file

The cache file is defined by the BufEvtPath statement in the configuration member pointed to by the EIF_CONF statement. The default cache file name is /etc/Tivoli/tec/cache_itepevent. For more information about the BufEvtPath statement and the EIF_CONF statement, see the *Event Pump for z/OS Configuration Reference*.

If you plan to validate SOAP requests (using the VALIDATE_SOAP_USERS start up parameter), implement the RACF PASSTICKET function:

1. Activate the PTKTDATA class:
`SETOPTS CLASSACT(PTKTDATA)`
`SETOPTS RACLIST(PTKTDATA)`
`RACLIST (PTKTDATA) REFRESH`
2. Define GPZPUMP (or the GTMPUMP) profile in the PTKTDATA class:
`RDEFINE PTKTDATA GTMPUMP SSIGNON(KEYMASKED(nnnnnnnnnnnnn))`
`SETOPTS REFRESH RACLIST(PTKTDATA)`

Chapter 6. Preparing the Event Pump program

The DEFLOGS member in the STGMSAMP data set allocates log data sets that are used by Event Pump.

Allocate the data sets as follows:

1. Review the job statement and modify it as required for your installation.
2. Review the data set characteristics including the amount of space to allocate.
3. Run the DEFLOGS job.
4. Verify your return codes.

After Event Pump starts, the data set DCB parameters are set to these values:

- RECFM is set to V
- LRECL is set to 32756
- BLKSIZE is set to 32760

Chapter 7. Installing and configuring the EIF probe

The following Tivoli Event Pump components are used for z/OS event processing on a Netcool/OMNIBus server:

- Tivoli EIF probe
- Schema
- EIF probe rules

The Tivoli EIF probe for Netcool/OMNIBus version 7.2.1, schema, and probe rules are located on the *IBM Tivoli Event Pump for z/OS Probe Rules DVD*, LCD7-3512. The EIF probe that is included on the DVD is only compatible for Netcool/OMNIBus version 7.2.1. To install the EIF probe for Tivoli Netcool/OMNIBus version 7.3.0 and higher, use the 4.2.2.1-TIV-EP-FP0001 package.

Before you install these components, review the following installation prerequisites and update the Netcool/OMNIBus server as required:

1. The Netcool/OMNIBus server where you plan to install the EIF probe has the Probe Support feature and a Java Runtime Environment.
2. The `NCHOME=install_dir` environment variables is set.

For information about the Probe Support feature, see the *IBM Tivoli Netcool/OMNIBus Installation and Deployment Guide*.

To install the Tivoli EIF probe, schema, and probe rules, follow these procedures:

- “Installing the EIF probe (Tivoli Netcool/OMNIBus 7.2.1 or earlier)” or “Installing the EIF probe (Tivoli Netcool/OMNIBus 7.3.0 or later)” on page 24
- “Copying files from the installation package” on page 25
- “Installing the schema updates” on page 25
- “Configuring the Netcool/OMNIBus ObjectServer” on page 26
- “Installing and configuring the EIF probe rules” on page 27

Installing the EIF probe (Tivoli Netcool/OMNIBus 7.2.1 or earlier)

In a Windows environment, follow these steps to install the EIF probe:

1. Insert the DVD into the drive. The LaunchPad starts automatically.
2. Click **Run the Tivoli EIF probe installation program** and follow the instructions for probe installation.

In a UNIX or Linux environment, follow these steps to install the EIF probe:

1. To start the LaunchPad, issue the following command from the root directory of the DVD:
`launchpad.sh`
2. Click **Run the Tivoli EIF probe installation program** and continue with the probe installation.

Usage note for Linux on zSeries®: Upload the `probe-zlinux.bin` executable file from the `bin` directory of the DVD. After you upload the file, run the `probe-zlinux.bin` file to install the Tivoli EIF probe.

Installing the EIF probe (Tivoli Netcool/OMNIBus 7.3.0 or later)

Starting with the Netcool/OMNIBus V7.3.0 product, all probes are supplied as packages. You can download each package from the IBM Passport Advantage® website.

Follow these steps to install the EIF probe:

1. Download the installation package for the probe from the Passport Advantage website.

Each probe has an installation package for each supported operating system. For details about how to locate and download the installation package for your operating system, see the Tivoli Information Center website at http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_OMNIBus.doc/probes/install/wip/out-html/reference/install_download_intro.html.

2. Install the probe using the installation package.

The installation package contains the appropriate files for all supported versions of the Netcool/OMNIBus product. For details about how to install the probe to run with your version of the Netcool/OMNIBus product, see the Tivoli Information Center website at http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_OMNIBus.doc/probes/install/wip/out-html/reference/install_install_intro.html.

In a Windows environment, follow these steps to install the EIF probe:

1. Download the Windows installation package for the Tivoli EIF probe from the Passport Advantage website at http://www.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm
2. Extract the contents of the package to a temporary location.
3. Run the following command to start the wizard installer:
`%NCHOME%\omnibus\install\nci_install_integration`
4. Accept the license terms.
5. The installer prompts you for the location of the probe. Enter the temporary extract directory.
6. Complete the installation.
7. Run the following command to start the console installer:
`%NCHOME%\omnibus\install\nci_install_integration -i console`

Usage note: To run the silent installer, follow these steps:

1. Create a file named `response.txt` with the following contents:
`LICENSE_ACCEPTED=true`
`PROBE_OR_GATE_LOCATION<ABSOLUTE PATH to temporary directory>`
2. Run the following command:
`%NCHOME%\omnibus\install\nci_install_integration -i silent -f <ABSOLUTE PATH to temporary directory>`

In a UNIX or Linux environment, follow these steps to install the EIF probe:

1. Download the UNIX or Linux installation package for the Tivoli EIF probe from the Passport Advantage website at http://www.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm
2. Extract the contents of the package to a temporary location.
3. Run the following command to start the wizard installer:

```
%NCHOME%\omnibus\install\nci_install_integration
```

4. Accept the license terms.
5. The installer prompts you for the location of the probe. Enter the temporary extract directory.
6. Complete the installation.
7. Run the following command to start the console installer:

```
$NCHOME/omnibus/install/nci_install_integration -i console
```

Usage note: To run the silent installer, follow these steps:

1. Create a file named `response.txt` with the following contents:

```
LICENSE_ACCEPTED=true  
PROBE_OR_GATE_LOCATION<ABSOLUTE PATH to temporary directory>
```

2. Run the following command:

```
$NCHOME/omnibus/install/nci_install_integration -i silent -f  
<ABSOLUTE PATH to temporary directory>
```

For additional installation and configuration information, see the *IBM Tivoli Netcool/OMNIBus Probe for Tivoli EIF and Tivoli Netcool/OMNIBus Event Integration Facility Reference* .

Copying files from the installation package

Before you update the Netcool/OMNIBus schema or install the EIF probe rules, copy the required files from the Tivoli Event Pump 4.2.2 Feature Pack 1 installation package to a download directory on the Netcool/OMNIBus EIF Probe server. Extract the contents of the 4.2.2.1-TIV-EP-FP0001 package. This action creates the 4.2.2.1-TIV-EP-FP0001 directory in your current download directory.

In a Windows environment, change to the 4.2.2.1-TIV-EP-FP0001 directory, and run the `zOSEventSupport_copy.bat` file to copy the files.

In a UNIX or Linux environment, change to the 4.2.2.1-TIV-EP-FP0001 directory, and run the `zOSEventSupport_copy.sh` file to copy the files.

The files are copied to the EventPump directory, under the `%NCHOME%\omnibus\etc` directory for the Windows environment or the `%NCHOME%\omnibus/etc` directory for a UNIX or Linux environment. If the EventPump directory does not exist, the directory is created.

Installing the schema updates

The Netcool/OMNIBus component for z/OS support includes the following updates to the ObjectServer components:

- Additional columns for z/OS-specific data
- A lookup table for setting the message severity and state for z/OS events
- Triggers for performing the lookup function and for processing message and resolution (clearing) events

In a Windows environment, follow these steps to install the schema updates for z/OS events on a Netcool/OMNIBus ObjectServer:

1. On the Netcool/OMNIBus server, change to the `%NCHOME%\omnibus\etc\` EventPump directory.

2. Update the following parameters in the `zOSEventSupport_sql.bat` file for your environment:

server Netcool/OMNIBus ObjectServer

user A user ID to log on to the indicated server

password

A password to log on to the indicated server

sql An SQL file (`zOSEventSupport.sql`) that must be run

This batch file attempts to connect to the indicated object server, using the supplied user ID and password.

3. Run the `zOSEventSupport_sql.bat` file. The Netcool/OMNIBus ObjectServer must be started to apply these schema changes.

In a UNIX or Linux environment, follow these steps to install the schema updates for z/OS events on a Netcool/OMNIBus ObjectServer:

1. On the Netcool/OMNIBus server, change to the `$NCHOME/omnibus/etc/EventPump` directory.

2. Update these parameters in the `zOSEventSupport_sql.sh` file for your environment:

server Netcool/OMNIBus ObjectServer

user A user ID to log on to the indicated server

password

A password to log on to the indicated server

sql An SQL file (`zOSEventSupport.sql`) that must be run

Using the supplied user ID and password, the `.sh` file attempts to connect to the indicated object server.

3. Run the `zOSEventSupport_sql.sh` file. The Netcool/OMNIBus ObjectServer must be started to apply these component changes.

Configuring the Netcool/OMNIBus ObjectServer

To support the processing of z/OS events by the Netcool/OMNIBus ObjectServer, a modified version of the *deduplication trigger* (named `zos_deduplication`) was added when the `zOSEventSupport.sql` file updated the `alerts.status` table. This trigger contains a WHEN expression that allows it to only process events that are defined as z/OS events.

It is important to protect the standard deduplication trigger so that it does not process z/OS events. To do this, modify the deduplication trigger definition and add a WHEN expression to the deduplication trigger.

Figure 2 on page 27 illustrates the use of the Netcool/OMNIBus administrator to add this WHEN expression to the deduplication trigger:

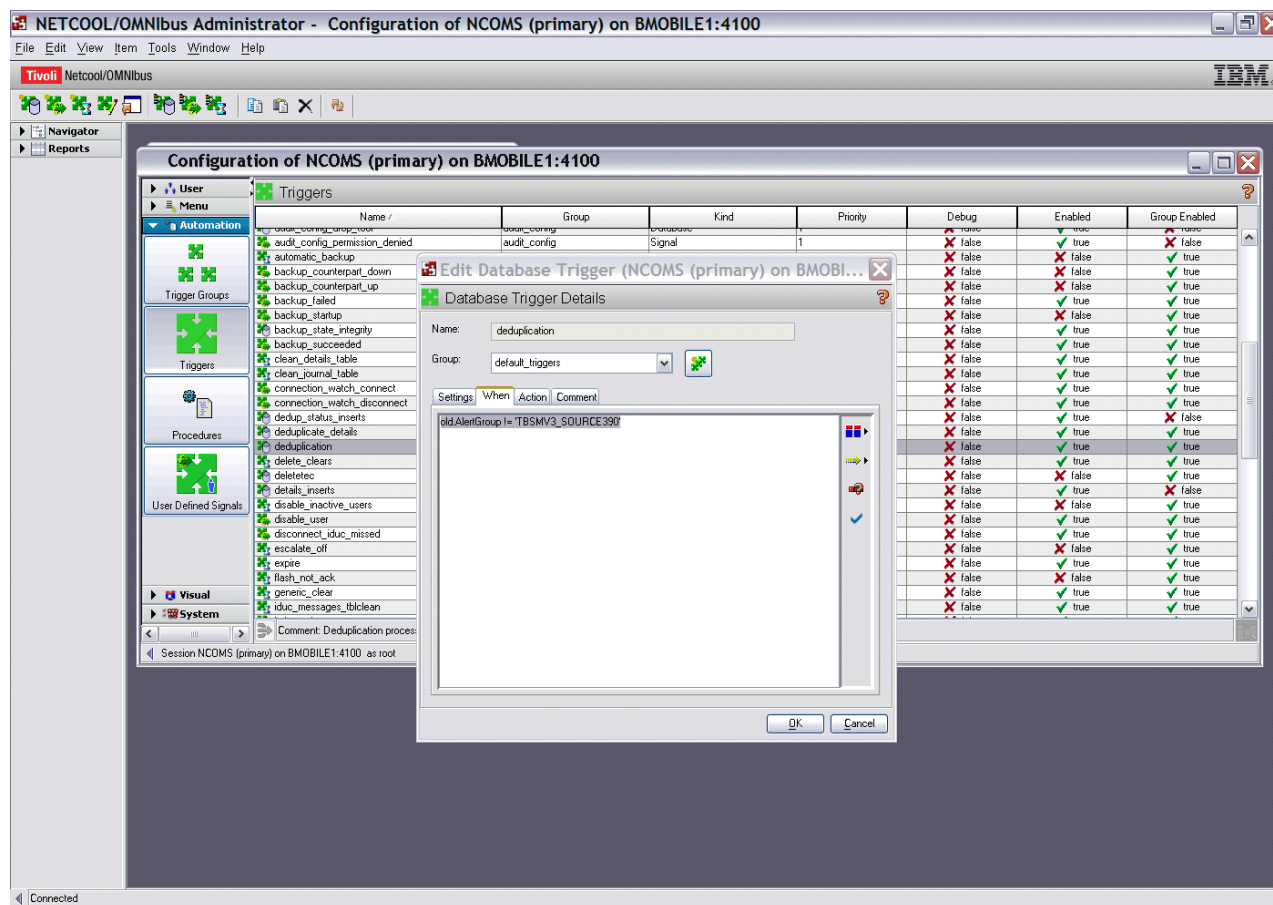


Figure 2. `old.AlertGroup != 'TBSMV3_SOURCE390'`

You can do additional administrative tasks (for example, updating the Message Event Severity Table). For information about these tasks, see Chapter 27, “Updating message event severity and state using Netcool/OMNIBus,” on page 151.

Installing and configuring the EIF probe rules

The Netcool/OMNIBus component for z/OS support includes EIF probe rules files. These rules files correlate events with z/OS Discovery Library Support updates.

Rule file include structure

The rules files that are used by the EIF probe support the inclusion of other rules files. The set of z/OS rules files uses this inclusion capability to organize and modularize the files. Include only the rules files that you need. By including only the rules files that you need, you reduce processor usage for the EIF probe.

Figure 3 on page 28 shows the hierarchical file structure for the rules file.

```

tivoli_eif.rules
    zos_event.rules
        zos_afoper_event.rules
        zos_autooperator_event.rules
        zos_cics_event.rules
        zos_customer_applications.rules
        zos_db2_event.rules
        zos_extension_event.rules
        zos_generic_event.rules
        zos_ims_event.rules
        zos_jobmon_event.rules
        zos_mvpf_event.rules
        zos_opsmvs_event.rules
        zos_rmf_event.rules
        zos_sa390_event.rules
        zos_sms_event.rules
        zos_tws_event.rules
        zos_zos_event.rules
    zos_event_user_defined.rules
    customer_override.rules

```

Figure 3. Rules file include structure

The indentation within the hierarchical view represents where files are included. For example, the `tivoli_eif.rules` file includes the `zos_event.rules` file and the files that are listed under it.

The following descriptions are for the rule files that are included in the `tivoli_eif.rules` file:

zos_event.rules

This rules file is the master z/OS probe rules file with include statements for other z/OS event source rules files.

zos_event_user_defined.rules

The individual rules file contain logic for mapping Event Pump events into Netcool/OMNIbus alerts.

customer_override.rules

This probe rules file contains rules that you might have for event processing. This rule file contains logic or mapping rules that can be used to replace or override any previous logic or mapping rules. When included, this file must always be the last file included in the `tivoli_eif.rules` file.

Installation steps

The probe rules are included for Windows systems, UNIX systems, and Linux systems (x86 and zSeries).

For Windows systems (Netcool/OMNIbus server files)

If you made any modifications to the rules files in the `%NCHOME%\omnibus\probes\win32` directory, make sure that you save a copy of these files. After the updated rules files are copied, merge your changes into the updated files.

Follow these steps to install and configure a system for z/OS events:

1. On the Netcool/OMNIbus server, change to the `%NCHOME%\omnibus\etc\EventPump` directory.
2. Run the `zOSEventSupport_rules.bat` file.

This batch file copies a set of rules files to the `%NCHOME%\omnibus\probes\win32` directory.

3. Change to the %NCHOME%\omnibus\probes\win32 directory.
4. Uncomment the following include statements in the tivoli_eif.rules file to enable the z/OS rules:

```
###
### Handle Z Events
###
include "zos_event.rules"
###
### Handle Z user defined events.
###
include "zos_event_user_defined.rules"
```

5. Stop and restart the probe to reread the new rules files.

For UNIX and Linux systems (Netcool/OMNIBus server files)

The rules files are located in the \$NCHOME/omnibus/probes/\$ARCH directory, where \$ARCH is defined as follows:

Operating system

	\$ARCH value
SunOS	solaris2
AIX®	aix4 or aix5
HP-UX	hpux10 or hpux11
Linux	linux2x86 or linux2s390

If you made any modifications to the rules files, make sure that you save a copy of these files. After the updated rules files are copied, merge your changes into the updated files.

Follow these steps to install and configure a system for z/OS events:

1. On the Netcool/OMNIBus server, change to the \$NCHOME/omnibus/etc/EventPump directory.
2. Run the zOSEventSupport_rules.sh script.
This script copies a set of rules files to the \$NCHOME/omnibus/probes/\$ARCH directory.
3. Change to the \$NCHOME/omnibus/probes/\$ARCH directory.
4. Uncomment the following include statements in the tivoli_eif.rules member to enable the z/OS rules:

```
###
### Handle Z Events
###
include "zos_event.rules"
###
### Handle Z user defined events.
include "zos_event_user_defined.rules"
```

5. Stop and restart the probe to reread the new rules files.

Configuration steps

After you install the EIF probe rules for z/OS, there are no additional required configuration steps. However, you can comment or uncomment individual rule file include statements in the zos_event.rules file to select specific z/OS data sources.

Chapter 8. Getting ready to start the Event Pump program

The Event Pump uses the following address spaces:

- Master
- Data space

Before configuring the Event Pump procedures, copy the following startup procedures for the address spaces from the SGTMSAMP library to a user data set:

GPZPUMP Master startup procedure

GPZDSPC (GTMSAM07)
Data space startup procedure

Make any modifications to the procedures in the user data set. By using the user data set, you protect your modifications when maintenance is applied.

Each startup procedure uses one or more startup parameter members. Copy the following startup parameter members from the SGTMSAMP library to a user data set:

GPZPARMS
Master startup parameter member

PARMDSPC (GTMSAM10)
Data space startup parameter member

PARMEIF (GTMSAM24)
EIF configuration file

PRMEDRSL (GTMSAM25)
Event driven registration (EDR) event file

PRMEIFSL (GTMSAM23)
EIF select table

REGDATA (GTMSAM22)
Registration data statements

Figure 4 on page 32 shows the primary input and output data files that are used by the master address space.

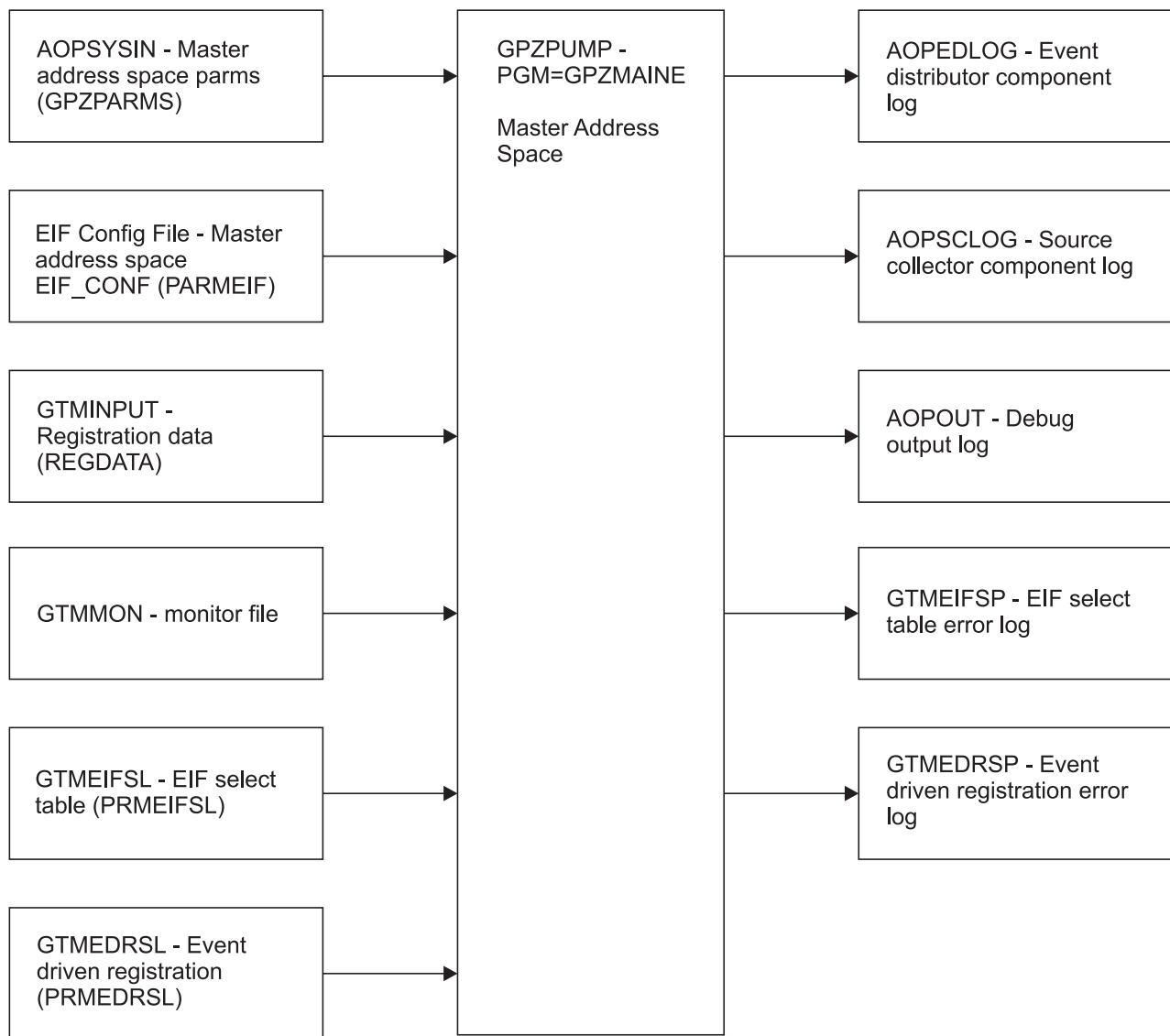


Figure 4. Master address space data files

Updating the GPZPUMP startup procedure

Update the master startup procedure member (GPZPUMP) in the user data set as required for your environment:

1. Update the following high-level qualifiers and associated data set names:
 - Tivoli Event Pump data sets (%hlq)
 - REXX alternate runtime library (%rexhlq)
 - CICSplex System Manager authorized library (%cpsmlib)
2. Change the data set name of the SGTMSAMP data set to specify the data set to which you copied the Event Pump startup parameter members.
3. Optional: If you plan to use Event Pump to monitor CICSplex/SM managed CICSplexes, uncomment the CPSM authorized library:

```
//STEPLIB DD DISP=SHR,DSN=%h1q.SGTMMODS <== AUTH LIB
// DD DISP=SHR,DSN=%rexh1q.SEAGALT <== AUTH LIB
//* DD DISP=SHR,DSN=%cpsmlib.SEYUAUTH <== AUTH LIB
```

If you plan to use the Event Integration Facility (EIF) for filtering, and the SYS1.SCREERUN library is not available through your link-list concatenation, add this library to the STEPLIB DD concatenation. In this case, the library must be APF-authorized.

4. The AOPSYSIN DD statement specifies the GPZPARMS member that contains the startup parameters for the master startup procedure:

```
//AOPSYSIN DD DISP=SHR,DSN=%h1q.SGTMSAMP(&MBR)
```

Make sure that the AOPSYSIN DD statement specifies the user data set that you created.

Review and update the following required parameters:

- COMMAND_PREFIX
- CONSOLE
- EIF_CONF

This statement can refer to an HFS file or a Partitioned Data Set (PDS) member

- LOG1

Use the data set names that you specified when you ran the DEFLOGS job.

- LOG2

Use the data set names that you specified when you ran the DEFLOGS job.

- SUBSYSTEM_NAME

If you are using the EDI function, you must specify a subsystem name (the subsystem must be active). If you run multiple Event Pump programs on a single z/OS LPAR, each Event Pump program must specify and use its own subsystem name.

- TBSM
- TCPIP_JOBNAME, if required by your installation

For more information on these parameters, see *Event Pump for z/OS Configuration Reference*.

Most startup parameters are optional and have default settings. Review and update the parameters that are described in Table 53 on page 155.

5. Registration data statements indicate which events Event Pump gathers and forwards to the EIF probe. The GTMINPUT DD statement defines the REGDATA (GTMSAM22) member that contains the data registration parameters that are used by Event Pump:

```
//GTMINPUT DD DISP=SHR,DSN=%h1q.SGTMSAMP(&REGMBR)
```

Make sure that the GTMINPUT DD statement specifies the user data set that you created.

Review the GLOBAL statements in the REGDATA member and customize them as needed. For information about the registration data statements, see Chapter 23, “Configuring registration data (REGDATA),” on page 133.

6. Optional: If you plan to use the Event Pump customizer to define user events (specific messages that you want to trap) or if you plan to use the event feeds listed in Table 1 on page 34, uncomment the GTMMON statement:

```
//*GTMMON DD DISP=SHR,DSN=%h1q.GTMMON
```

The GTMMON DD statement identifies the monitor file that implements the definitions necessary to monitor messages that are produced by the subsystem. For information on how to define messages, see Chapter 26, “Creating user-defined events,” on page 143.

Notes:

- a. The GTMMON library is a partitioned data set with 80-byte records.
- b. The SGTMSAMP library contains sample GTMMON members that you can use without modification. To use these samples, add the SGTMSAMP library to the GTMMON DD statement concatenation.

Table 1. GTMMON samples

Event feed	Sample
BMC MainView for CICS	MCICPARM (GPZMMCIC)
BMC MainView for DB2	MDB2PARM (GPZMMDB2)
BMC MainView for z/OS	MZOSPARM (GPZMMZOS)
CICSplex System Manager	ICICPARM (GPZMCPSM)
DB2	IDB2PARM (GPZMDB2)
Health Checker	IHCKPARM (GPZMHCK)
Resource Management Facility (RMF™)	
- default exception	RMFDEFLT (GPZRMFDE)
- stress exception	RMFSTRES (GPZRMFST)
- customized exception	GPZRMFCU
Storage Management Resource Monitoring (SMS)	
- messages	ISMSPARM (GPZMSMS)
- cache performance exceptions (default)	CPEDEFLT (GPZCPEDE)
- cache performance exceptions (stress)	CPESTRES (GPZCPEST)
- device performance exceptions (default)	DPEDEFLT (GPZDPEDE)
- device performance exceptions (stress)	DPESTRES (GPZDPEST)
- space usage exceptions (default)	DSEDEFLT (GPZDSEDE)
- space usage exceptions (stress)	DSESTRES (GPZDSEST)

7. Optional: The Dynamic Trap Local Data Store (DTLDS) is a sequential data set that holds trap creation data that must be retained across Event Pump restarts. Each Event Pump instance has its own DTLDS data set. Uncomment the GTMDTLDS DD statement if you plan to enable sysplex support or use Event Driven Registration:

```

//*GTMDTLDS DD DSN=%hlq.GTMDTLDS,DISP=MOD,
//*             DCB=(LRECL=1024,BLKSIZE=0,RECFM=VB),
//*             UNIT=SYSDA

```

Usage note: Make sure that the DTLDS data set is allocated with a variable record (VB) format and that it has a minimum record size of 1024 bytes.

8. If TCP/IP domain name resolution is not fully configured at your site, uncomment the SYSTCPD DD statement and update the TCPDATA library name:

```

//*SYSTCPD DD DISP=SHR,DSN=%tcphlq.SEZAINST(TCPDATA)

```

9. If your installation runs more than one TCP/IP stack, the Event Integration Facility (EIF) component must gain affinity to the proper stack using one of the following methods:

- The BPXTCAFF program can be used to bind affinity to a stack of your choice. The BPXTCAFF program must run before the program that initializes the address space or before the program that initializes the address space containing the Event Integration Facility.

Depending on your installation, run the BPXTCAFF program before the master address space. One way of doing this is to add this statement in the Tivoli Event Pump startup JCL before the statement that is used to start the Event Integration Facility:

```
//STEP0 EXEC PGM=BPXTCAFF,PARM=MYSTACK
```

where the PARM parameter is a pointer to the stack to which you want to establish affinity.

This example shows how the startup JCL might look after the step is added:

```
//STEP0 EXEC PGM=BPXTCAFF,PARM=MYSTACK
//PUMP EXEC PGM=GPZMAINE,TIME=1440,REGION=400M
//STEPLIB DD DISP=SHR,DSN=%h1q.SGTMMODS <== AUTH LIB
// DD DISP=SHR,DSN=%rexh1q.SEAGALT <== AUTH LIB
// DD DISP=SHR,DSN=%cpsmlib.SEYUAUTH <== AUTH LIB
```

- To request transport affinity when multiple TCP/IP images are used, set the BPXK_SETIBMOPT_TRANSPORT environment variable to the name of the preferred transport before starting the Event Integration Facility (GPZPUMP started procedures).

For additional information about the BPXTCAFF program and the BPXK_SETIBMOPT_TRANSPORT environment variable, see *UNIX System Services Planning*.

10. Event messages can be intercepted within the event distributor component as they are sent to the EIF probe using EIF. You can filter these z/OS events by using an EIF select table. The EIF select table name (PRMEIFSL by default) is specified by the GTMEIFSL DD statement:

```
//GTMEIFSL DD DISP=SHR,DSN=%h1q.SGTMSAMP(&EIFSLMBR)
```

Make sure that the GTMEIFSL DD statement specifies the user data set that you created.

The PRMEIFSL member in the SGTMSAMP data set contains sample EIF table statements. To set up message selection and filtering, review the contents of the PRMEIFSL member and change it as necessary for your installation. For information about specifying the EIF select table, see Chapter 24, “Filtering Event Pump for z/OS events,” on page 137.

The GTMEIFSP DD statement is used to report any errors with SELECT statements in the PRMEIFSL member:

```
//GTMEIFSP DD SYSOUT=X
```

11. Optional: You can use the Event-Driven Registration (EDR) function to dynamically register messages based on other messages that are currently registered. You can define these messages within an EDR event file. To use the event-driven registration function, follow these steps:

- Uncomment the following GTMEDRSL and GTMEDRSP DD statements:

```
//*GTMEDRSL DD DISP=SHR,DSN=%h1q.SGTMSAMP(&EDRSLMBR)
//*GTMEDRSP DD SYSOUT=X
```

Make sure that the GTMEDRSL DD statement specifies the user data set that you created.

- Modify the PRMEDRSL member as required for your installation. For information about how to update this member, see Chapter 25, “Configuring event-driven registration,” on page 141.
12. If you plan to use multiple instances of Event Pump, each instance and its associated data space must include the following statement:

```
//ACC1IDxx DD DUMMY
```

where *xx* is a 2-character suffix that defines the instance. If you omit this DD statement, the ID that is used by Event Pump defaults to 01.

For additional information about modifying the GPZPUMP procedure, see the procedure prolog.

Updating the EIF configuration file

The name of the EIF configuration file is specified by the EIF_CONF statement in the GPZPARMS member of the SGTMSAMP data set. By default, the EIF configuration file is the PARMEIF (GTMSAM24) member.

To update the EIF configuration file, modify the statements in the PARMEIF member as required for your installation. The ServerLocation and ServerPort statements are required and must match the host name of the Netcool® OMNIbus object server and the port on which the EIF probe listens. Refer to the Netcool OMNIbus library on configuring the object server and EIF probe.

The other statements are optional and have default settings if not specified.

If you want information about...	Refer to...
EIF configuration file statements	<i>Event Pump for z/OS Configuration Reference</i>

Updating the GPZDSPC startup procedure

Update the GPZDSPC data space startup procedure member in the user data set as required for your environment:

1. Update the high-level qualifier (%hlq) for the Event Pump data sets:

```
//STEPLIB DD DISP=SHR,DSN=%hlq.SGTMMODS
//ACC1PARM DD DISP=SHR,DSN=%hlq.SGTMSAMP
```
2. Change the data set name of the SGTMSAMP data set to specify the data set to which you copied the Event Pump startup parameter members.
3. The DSPC EXEC statement specifies the PARMDSPC member which contains the parameters for the data space startup procedure:

```
//DSPC EXEC PGM=GTMACC17,PARM=&MBR
```

You can customize the LOGQUEUE SIZE startup parameter. For information about startup parameters, see *Event Pump for z/OS Configuration Reference*.

4. If you plan to use multiple instances of Event Pump, each instance and its associated data space must include the following statement:

```
//ACC1IDxx DD DUMMY
```

where *xx* is a 2-character suffix that defines the instance. If you omit this DD statement, the ID that is used by Event Pump defaults to 01.

5. Copy the procedure to your PROCLIB data set.

For additional information about modifying the GPZDSPC procedure, see the procedure prolog.

Chapter 9. Verifying the Installation

To test the Event Pump installation, follow these steps:

1. "Verify that EIF logging is enabled"
2. "Starting and stopping Event Pump procedures"
3. "Review the AOPEDLOG log" on page 40

Verify that EIF logging is enabled

Make sure that EIF logging is enabled in the GPZPARMS member:

EIF_LOG=Y

Starting and stopping Event Pump procedures

To monitor a z/OS operating system, the following Event Pump address spaces must be running on the operating system that is monitored:

- Data space
- Master address space

Run the components as started tasks or batch jobs. Start the address spaces in the following order:

1. Data space
2. Master address space

To stop the data space, stop the address spaces in the following order:

1. Master address space
2. Data space

When you restart the data space, start the address spaces in the same start order:

1. Data space
2. Master address space

Starting and stopping the data space

To start the data space as a started task (default sample procedure), enter the following command at the system console:

```
S GPZDSPC [,MBR=data_space_name]
```

In this example, *data_space_name* is the name of a member in the library pointed to by the ACC1PARM DD statement in the data space started task procedure. The default name is PARMDSPC. This member contains the data space startup parameters.

After starting the primary data space, the GTM5010I message is displayed that indicates that data space initialization is complete. If you start a second copy of the data space to serve as a backup, the GTM4601I message indicates that the backup data space is monitoring the primary data space address space. The second copy of the data space can take over if a failure occurs.

Stop the master address space before you stop the data space. The data space does not shut down when the STOP (P)

command is used unless the master address space is stopped.

To stop the data space, enter the following command at the system console:

```
P data_space_name
```

Starting and stopping the master address space

To start the master address space as a started task (if the default sample procedure is used), enter the following command at the system console:

```
S GPZPUMP [,MBR=master_address_space_name]
```

In this example, *master_address_space_name* is the name of a member in the library pointed to by the AOPSYSIN DD statement in the master started task procedure. The default name is GPZPARMS. This member contains the master startup parameters. The GTM0990I message is displayed after starting the master address space and indicates that data space initialization is complete.

You can stop and start the master again without stopping and restarting the data space.

If the SHUTDOWN_CONFIRM startup parameter is set to YES, reply SHUT to the GTM9000I message.

To stop the master, enter the following command at the system console:

```
P master_address_space_name
```

Review the AOPEDLOG log

Review the AOPEDLOG log. In the sample procedure, the AOPEDLOG log is a SYSOUT data set.

The AOPEDLOG log displays the startup parameters that were processed, followed by events that were generated by Event Pump. Each event is identified by the EIF0301 message. The event contents follow this message. The event is sent to the event server that you specified when you updated the EIF configuration file.

Figure 5 shows an example of a generated event.

```
EIF0301 : TEC EVENT DATA FOLLOWS                                TBSM
TBSMV3_SOURCE390;source=USIBMNT.NTEFMVS;name_valueTBSM
_pairs=|'_FORMAT=01','_ACTION=01','_F00=2011090713TBSM
:19:54.090099','_F01=01611','_F02=0S','_F03=0161',TBSM
'_F73=08/29/2011','_F74=08:43:58','_F75=9.42.45.16TBSM
1','_F76=NMP161','_F77=USIBMNT.NTEFMVS','_F81=PLEXTBSM
1';END                                                            TBSM
EIF0302 : END OF TEC EVENT DATA                                TBSM
EIF0303 : CALLING EIF TO SEND EVENT TO TEC                      TBSM
EIF0304 : EIF RC=00000000                                        TBSM
```

Figure 5. AOPEDLOG log example

Part 3. Enabling data sources and event feeds

Chapter 10. Data source and event feed summary	45
Chapter 11. Data sources	47
JOBLOG	47
Specifying JES SPOOL data sets that you want to monitor	47
Adding traps for the JOBLOG data source	49
Registering JES JOBLOG as a message source	49
JES3 environment restrictions	49
SYSLOG	50
Specifying SYSLOG messages that you want to monitor	50
Registering the SYSLOG as a message source	51
Chapter 12. Automation	53
Tivoli AF/OPERATOR	53
Installing the AF/OPERATOR EDI interface	53
Reviewing the Event Pump GPZPARMS member	54
Customizing AF/OPERATOR to send events	54
Calling the EDI	54
Starting and ending the automation process	54
Additional modification examples	55
Error handling	56
Modifying the source	56
Tivoli NetView for z/OS automation	57
Tivoli System Automation for z/OS	58
Event flow	59
Installing System Automation for z/OS	59
Updating the System Automation for z/OS control file	59
Customizing the NetView program to enable Event Pump to process events	60
Specifying the resources to be monitored	60
Customizing System Automation for z/OS to send events	60
Calling the EDI	61
Starting and ending the automation process	61
Additional examples of requests	62
Error handling	62
Modifying the source	63
Restarting the NetView program	64
Verifying the installation	64
Disable logging the events that are sent to the PPI receiver	65
BMC MainView AutoOPERATOR for z/OS	65
Installing the AutoOPERATOR EDI interface	65
Reviewing the Event Pump GPZPARMS member	66
Customizing BMC MainView AutoOPERATOR to send events	66
Calling the EDI	66
Starting and ending the automation process	66
Additional modification examples	67
Error handling	68
Modifying the source	68
Computer Associates OPS/MVS	69
Installing the CA OPS/MVS EDI interface	70
Reviewing the Event Pump GPZPARMS member	70
Customizing the CA OPS/MVS event feed to send events	71
Calling the EDI	71
Starting and ending the automation process	71
Additional examples of requests	72
Error handling	73
Modifying the source	73

Chapter 13. BMC MainView	75
BMC MainView for z/OS.	75
Event flow.	75
Event scenario	76
Configuring BMC MainView for z/OS	76
Installing Event Pump for BMC MainView z/OS components	76
Registering BMC MainView for z/OS	76
Enabling event processing	76
Disabling event processing	76
BMC MainView for CICS.	77
Event flow.	77
Event scenarios	78
DB2 CICS interface unavailable.	78
DB2 CICS interface becomes available again	78
Configuring BMC MainView for CICS	79
Registering BMC MainView for CICS.	79
Enabling event processing	79
Disabling event processing	79
BMC MainView for DB2	79
Event flow.	80
Event scenarios	81
TSO user exceeds its processor resource usage threshold	81
TSO user processor resource use drops beneath threshold	81
Configuring BMC MainView for DB2.	81
Registering BMC MainView for DB2	82
Enabling event processing	82
Disabling event processing	82
 Chapter 14. CICS	 83
Installing the CICS Transient Data Queue Exit program	83
Review the CICS started task	84
Reviewing the Event Pump GPZPARMS member.	84
Registering base CICS resources and CICS Transient Data	84
 Chapter 15. CICSplex System Manager	 85
Reviewing resource definitions	85
Defining System Availability Monitoring Alerts	86
Defining MAS Resource Monitoring Alerts.	86
Excluding resources and events.	87
Reviewing the GPZPUMP procedure	88
Registering CICSplex System Management source	88
 Chapter 16. DB2 Universal Database for z/OS	 91
Installing Event Pump for DB2 components	91
Adding DB2 sysplex support	91
Review security	92
Configuring each DB2 subsystem or data sharing group	92
Reviewing the GPZPUMP procedure	93
Registering the DB2 event feed for monitoring.	93
Enabling DB2 event processing	93
Disabling DB2 event processing	93
Excluding resources and events.	94
Supported resource monitoring.	94
Reasons for excluding resources or events	94
How to exclude resources or events	95
Hierarchy of exclude processing	95
Refreshing exclude rules	96
Show a list of Exclude Rules.	96
Syntax of the Exclude Rules	96
 Chapter 17. Health Checker for z/OS.	 101

Registering Health Checker data feed for monitoring	101
Assigning Health Checker filtering and severities	101
Chapter 18. IMS	103
Installing the Event Pump for IMS components	104
Enabling the OTMA callable interface	104
Configuring the IMS security parameters	105
Installing the IHS\$MPF exit	105
Installing the IMS AO exit	106
Verify the AO exit installation	106
Determine coexistence issues with CA OPS/MVS	107
Configuring the program-to-program interface (PPI)	107
Configuring the NetView components	107
Verifying the installation	108
Verifying host customization	108
Verifying end-to-end connectivity.	108
Verifying IMS event monitoring	108
Verifying poll monitors	109
Disabling the logging of all Event Pump events to the NetView log	109
Customizing event processing	109
Customizing polling processing operation and thresholds	110
Excluding resources and events	112
Tuning for IMS monitoring	113
Chapter 19. Resource Management Facility (RMF)	115
Configuring RMF	115
Updating the master address space parameters (GPZPARMS)	115
Updating the registration data file (REGDATA)	115
Including RMF statements	116
Customizing RMF exception definitions	116
Chapter 20. Storage Management Resource Monitoring (SMS).	121
Registering SMS as a message source	121
Registering disk device and storage volume resources	122
Chapter 21. Tivoli NetView for z/OS	123
Installing Event Pump components	123
Determine the NetView for z/OS started tasks	123
Customize each NetView for z/OS started task	124
Updating NetView for z/OS started task JCL.	124
Configuring NetView parameter members.	124
Modifying security settings for the NetView program	126
Defining the program-to-program interface connection	127
Customize event feeds	128
Chapter 22. Tivoli Workload Scheduler for z/OS.	129
Update Event Pump to receive events from Tivoli Workload Scheduler	129
Update Tivoli Workload Scheduler	129

Chapter 10. Data source and event feed summary

Event Pump monitors data sources (SYSLOG and JOBLOG) and event feeds for conditions that cause an event to be sent to the event server that is specified in the PARMEIF member. To enable this processing, Event Pump must be configured to expect data from the event sources and in some cases to examine the operating system environment for conditions that indicate events. After configuring Event Pump to expect event data, in some cases the event source must be configured to send the events.

The GTMINPUT DD statement in the GPZPUMP startup procedure specifies registration statements that define the classes of events to be processed. These events in some cases are defined within the Event Pump product, for example DB2 and IMS. In other cases, the events are defined generically, with the specifics included in the data set that is specified by the GTMMON DD statement. In a few cases, such as automation, there is additional specification required to identify the automation environment. This additional specification is included in the Event Pump parameter member that is specified by the AOPSYSIN DD statement.

When a class of events is defined, events can be filtered using an EIF SELECT table contained in the data set that is specified by the GTMEIFSL DD statement.

You can configure Event Pump to handle pairs of events such as status-up and status-down conditions. This is done by specifying event driven registration using the data set that is specified by the GTMEDRSL DD statement.

Usage note: In a 3-address-space environment, use the GTMPUMP startup procedure instead of the GPZPUMP procedure.

Table 2 provides an overview of the configuration statements and their locations for each event feed

Table 2. Event feed or data source specifications

Data source or event feed	Config specs in AOPSYSIN (GPZPARMS)	Registration statements in GTMINPUT (REGDATA)	Monitor specs in GTMMON	Filtering specs in GTMEIFSL	Event driven registration specs in GTMEDRSL	Event feed customization required	Page
Data source:							
JOBLOG	No	Yes	Yes	Yes	Optional	No	47
SYSLOG	No	Yes	Yes	Yes	Optional	No	50
Event feed:							
CICS							
Base CICS	Yes	Yes	No	Yes	No	No	83
CICS Transient Data Queue	Yes	Yes	No	Yes	No	Yes	83
CICSplex System Manager	No	Yes	Yes	Yes	No	Yes	85
DB2	No	Yes	Yes	Yes	No	Yes	91

Table 2. Event feed or data source specifications (continued)

Data source or event feed	Config specs in AOPSYSIN (GPZPARMS)	Registration statements in GTMINPUT (REGDATA)	Monitor specs in GTMMON	Filtering specs in GTMEIFSL	Event driven registration specs in GTMEDRSL	Event feed customization required	Page
Health Checker for z/OS	No	Yes	Yes	Yes	No	No	101
IMS	Yes	No	No	Yes	No	Yes	103
RMF	Yes	Yes	Yes	Yes	No	No	115
Storage (SMS)	No	Yes	Yes	Yes	No	No	121
Tivoli							
AF/Operator	Yes	No	No	Yes		Yes	53
System Automation for z/OS, NetView for z/OS	Yes	No	No	Yes		Yes	58
Tivoli Workload Scheduler	Yes	No	No	Yes		Yes	129
BMC							
MainView AutoOPERATOR for z/OS	Yes	No	No	Yes	No	Yes	65
MainView for CICS	No	Yes	Yes	Yes	No	Yes	77
MainView for DB2	No	Yes	Yes	Yes	No	Yes	79
MainView for z/OS	No	Yes	Yes	Yes	No	Yes	75
Computer Associates							
OPS/MVS	Yes	No	No	Yes	No	Yes	69
User-defined	No	Yes	Yes	Yes	Optional	No	143

Chapter 11. Data sources

Event pump uses the following data sources to process messages:

- “JOBLOG”
- “SYSLOG” on page 50

Different event feeds use these data sources to gather information, including custom event processing that is described in Chapter 26, “Creating user-defined events,” on page 143.

JOBLOG

You can use the JES JOBLOG file as a source of messages when you process messages through matching traps. Every job writes data to the JES SPOOL data sets during the lifetime of the application. For example, you might want to monitor a number of major applications that write to a specific application log that includes important state and status information and you want this state and status information to be available in the OMNibus or TBSM products. To do this, use the Event Pump JOBLOG function. The main purpose of the JOBLOG function is to monitor specified JES SPOOL data sets and to trap messages from these data sets. After the message is trapped, it is sent to OMNibus ObjectServer like any other traps in the Event Pump architecture.

Table 3 provides an overview of the tasks that are required to configure Event Pump to monitor JES SPOOL data sets.

Table 3. Implementing JES JOBLOG function

Task	Where to find information
Select JES SPOOL DD statement to monitor	“Specifying JES SPOOL data sets that you want to monitor”
Specify traps for a JOBLOG data source	“Adding traps for the JOBLOG data source” on page 49
Register the JOBLOG file as a message source	“Registering JES JOBLOG as a message source” on page 49

Specifying JES SPOOL data sets that you want to monitor

You can customize the JES JOBLOG function to use the JES SPOOL data sets as a source of messages by using the configuration member in the *config_member* data set member that you specified in the REGDATA member. The configuration member is specified in the concatenation of the GTMMON DD statement in the GPZPUMP procedure.

Usage note: In a 3-address-space environment, the source collector startup procedure is GTMPUMP.

The configuration member has the following format:

```
JOBLOG JOBNAME=jobname      *
      DDNAME=ddname         *
      [JOBTYPE=STC,TSU,JOB]  *
      [JOBID=jobid]          *
      [SYSNAME=sysname]     *
```

[CLASS=class]	*
[PROCNAME=procname]	*
[STEPNAME=stepname]	*
[USERID=userid]	*
[LAYOUT=layout]	

For information on the JOBLLOG configuration statements, see *Event Pump for z/OS Configuration Reference*.

All JOBLLOG parameters support standard masks. This format requires a comment statement that is started from the asterisk symbol. Any data that is followed by an asterisk is considered a comment and is not part of the SPOOL DD description.

Declaration of the new SPOOL data set begins at the JOBLLOG statement. Place the declaration of the new SPOOL data set on the same line as the JOBLLOG statement. To merge lines, you can use any non-blank continuation character in position 72 of the configuration member. There are two mandatory parameters for the SPOOL DD statement: JOBLNAME and DDNAME. All other parameters are optional.

The definition for the JOBLNAME and DDNAME parameters do not adequately define a particular SPOOL data set in the JES JOBLLOG function. There might be several jobs with the same name and SPOOL DD statement name. In this case, the JOBLLOG manager automatically monitors all SPOOL data sets that are associated with a specified SPOOL filter. You can filter a SPOOL data set by changing optional parameters, for example STEPNAME, USERID, and other parameters. Sometimes, the characters that are at the beginning of a record in a SPOOL DD statement contain ordering information, such as a time stamp and job ID. Here is an example:

```
22.33.27 STC08784  EZZ6011I TELNET BPX1SOC FAILED, RC = 00000070 RSN = 12CA00B6
```

If you are interested in a string that starts from an event ID, set up a layout to specify the right data for matching. Each record from the SPOOL data set is superimposed on the layout. After this operation, the resulting string is matched to the trap. To specify the number of words that are skipped from the beginning of each record, specify a period (.) at the beginning of the layout. Each period represents one skipped word. There is a standard mask at the end of the layout that describes the simple matching condition for the current layout. If a mask is specified, then the JOBLLOG function traps all records that match the mask. All other records are skipped. The layout is applied to an entire SPOOL data set. You can specify only one layout for a particular SPOOL data set. The following table lists examples of layouts.

Table 4. Examples of layouts

Layout	Spool message	Superimposed message
. . EZZ*	22.33.27 STC08784 EZZ6011I TELNET BPX1SOC FAILED, RC = 00000070 RSN = 12CA00B6	EZZ6011I TELNET BPX1SOC FAILED, RC = 00000070 RSN = 12CA00B6
. GTM*	11.40.50 GTM5101I ITZP DATASPACE CREATED, SIZE= 42467328, ORIGIN=000	GTM5101I ITZP DATASPACE CREATED, SIZE= 42467328, ORIGIN=000

Here is an example of a configuration member:

```
*
*  Event Pump dataspace SPOOL DD monitoring
*
JOBLLOG JOBLNAME=GTM43DSP
```

```

        DDNAME=JESMSGLG
*
*  TCPIP SPOOL DD monitoring
*
JOBLOG  JOBNAME=TCPIP
        DDNAME=CFGPRINT
        SYSNAME=TVT5105
        CLASS=A
        JOBID=STC08783
        JOBTYP=STC

```

In this example, the following event sources are monitored:

- JESMSGLG data set that is associated with the GTM43DSP job.
- CFGPRINT data set that is associated with the TCPIP started task. To further refine the started task characteristics, the task runs on a system with an ID of TVT5105, job ID of STC08783, and job class A.

Adding traps for the JOBLLOG data source

After you specify the JES SPOOL DD statement that you want to monitor, use the Event Pump customizer to define user events, which are the specific messages that you want to trap. Use a TYPE JES statement in a ZEVENT entry to specify the traps. See Chapter 26, “Creating user-defined events,” on page 143 for information about ZEVENT statements.

Here is an example of a ZEVENT entry:

```

*
*  GTM5405I ITZP PARTNER ON 9.42.62.104 PORT 8082 DETECTED
*
ZEVENT GTM5405I*
  MASK %S*
  EVENTID GTM5405I
  RESOURCE GTM94PUM
  RESOURCE_TYPE STC
  SEVERITY_CLEAR
  TYPE JES
  MESSAGE $MESSAGE_TEXT
END

```

In this example, an event is generated when a message begins with the string GTM5405I. The event specifies a resource name of GTM94PUM, a resource class of STC, an event type of JES, and the message text as the value of the MESSAGE attribute. The event is defined as a clearing event.

Registering JES JOBLLOG as a message source

To set up the JOBLLOG JES SPOOL data sets for use as a message source, add the following statement to the GTMINPUT member that is used for JOBLLOG monitoring:

```

SOURCE TYPE=JOBLLOG
        CONFIG=config_member
        DEBUG=[YES|NO]

```

The asterisk (*) must be in column 72.

JES3 environment restrictions

Consider the following restrictions in a JES3 environment:

- Started tasks console (STC) messages are the same as batch job console messages. As a result, Event Pump cannot distinguish between a message from

an STC and one from a batch job. If an administrator adds batch jobs and STCs to an operating system with identical names under JES3, unpredictable results can occur.

- The JES3 environment cannot distinguish between resource types. Because of this, Event Pump sends the IEF452I message to the JES3 resource. You can resolve this issue by specific handling of the IEF452I message. This specific handling can also occur in JES2 environments.

One of the following actions occurs when you receive the IEF452I JCL error message:

- If the resource is found in Event Pump, the event message is posted to that resource.
- If the resource is not found, or the resource is identified as a duplicate name, Event Pump does not know where to post the event.
 - A Generic Object resource named JCL ERRORS FOR IEF452I ON *<osname>* is automatically generated under the operating system that received the message. This resource is created when the first IEF452I message is received. After the Generic Object resource is created, an exception message is posted to this resource identifying the JCL error that occurred.
 - If the Generic Object resource is already created, it is reused to continue to post exception messages for these IEF452I errors. An entry is added to the MVS_ErrorLog table to record the error. The error message provides the explanation why the exception was posted to the Generic Object resource. The MVS_ErrorLog table retains the most recent 3 days of errors.

SYSLOG

You can use SYSLOG monitoring to generate events for a user-defined resource based on messages written to the system log.

Table 5 provides an overview of the tasks that are required implement SYSLOG monitoring.

Table 5. Implementing SYSLOG monitoring

Task	Where to find information
Specify SYSLOG messages	"Specifying SYSLOG messages that you want to monitor"
Register SYSLOG messages	"Registering the SYSLOG as a message source" on page 51

Specifying SYSLOG messages that you want to monitor

Identify the SYSLOG messages to be monitored as event sources. A message can contain fixed text and variable text.

In the following example, these messages are monitored:

- TEST001W *resource_name resource_type # additional_text*
- TEST002W *resource_name resource_type # additional_text*
- TEST003I *resource_name resource_type # additional_text*

Create a member (for example, SYSLOG01) in the data set that is specified by the GTMMON DD statement in the GPZPUMP startup procedure. This member contains ZEVEN statements that describe each message and how it is parsed to create an event. Figure 6 on page 51 shows the ZEVEN statements describing the

messages that are monitored.

```
*
* TEST00xx TEST MESSAGE # n FOR CUSTOMIZER
*
ZEVENT TEST001W
  MASK %S %S # %S* FOR CUSTOMIZER
  EVENTID TEST
  RESOURCE $1
  RESOURCE_TYPE $2
  SEVERITY WARNING
  INFORMATION YES
END
ZEVENT TEST002W
  MASK %S %S # %S* FOR CUSTOMIZER
  EVENTID TEST
  RESOURCE $1
  RESOURCE_TYPE $2
  SEVERITY WARNING
  INTERVAL 60
END
ZEVENT TEST003I
  MASK %S %S # %S* FOR CUSTOMIZER
  EVENTID TEST
  RESOURCE $1
  RESOURCE_TYPE $2
  SEVERITY CLEAR
END
```

Figure 6. Sample GTMMON member SYSLOG01

The messages are parsed as follows:

- TEST001W is an informational message with a severity of WARNING. The first token in the message is the *resource_name*. The second token in the message is the *resource_type*.
- TEST002W is a message with a severity of WARNING, which expires 60 seconds after it is received by the Netcool/OMNIBus ObjectServer. The first token in the message is the *resource_name*. The second token in the message is the *resource_type*.
- TEST003I is a clearing message. The first token in the message is the *resource_name*. The second token in the message is the *resource_type*.

Usage note: In a 3-address-space environment, the startup procedure is GTMPUMP.

Registering the SYSLOG as a message source

Add the following statements to the REGDATA member to register the resource class to be monitored:

```
GROUP SYSLOG
MONITOR TYPE=TEST CONFIG=SYSLOG01
GROUP END
```

The REGDATA member is in the data set specified by the GTMINPUT DD statement in the GPZPUMP startup procedure.

Enable the updated REGDATA member using the following z/OS command:

```
f gpzpump,register syslog
```

Usage note: In a 3-address-space environment, the startup procedure is GTMPUMP.

Chapter 12. Automation

The automation external data interface (EDI) enables automation applications to send events to Event Pump that are related to the resources that they manage. The EDI uses cross-memory services to pass messages or exceptions that reside on the same z/OS operating system. The EDI inserts a time stamp and sequence number on a message or exception, adjusts the final length, and sends the data to Event Pump. Messages and exceptions are passed directly to Event Pump by the EDI.

The following automation product event feeds integrate with Event Pump.

- “Tivoli AF/OPERATOR”
- “Tivoli NetView for z/OS automation” on page 57
- “Tivoli System Automation for z/OS” on page 58
- “BMC MainView AutoOPERATOR for z/OS” on page 65
- “Computer Associates OPS/MVS” on page 69

If you want information about...	Refer to...
----------------------------------	-------------

External data interface overview	Appendix D, “External Data Interface overview,” on page 173
----------------------------------	---

Tivoli AF/OPERATOR

The Event Pump interfaces with the IBM Tivoli AF/OPERATOR external automation product. With the implementation of the EDI, an external application program can pass text-based messages directly to Event Pump on the same MVS system.

Table 6 provides an overview of the tasks that are required to configure AF/OPERATOR to interface with Event Pump.

Table 6. Implementing the AF/OPERATOR event feed

Task	Where to find information
Install the AF/OPERATOR EDI interface.	“Installing the AF/OPERATOR EDI interface”
Review the GPZPARMS member and update as necessary.	“Reviewing the Event Pump GPZPARMS member” on page 54
Customize the resource monitoring levels for AF/OPERATOR resources.	“Customizing AF/OPERATOR to send events” on page 54

Installing the AF/OPERATOR EDI interface

Make the following modifications that are required for installing the AF/OPERATOR EDI interface:

1. Edit the AF/OPERATOR started task and make the following updates:
 - Add the `hlq.SGTMEEXEC` data set to the SYSEXEC DD concatenation.
 - Add the `hlq.SGTMMODS` data set to the STEPLIB DD concatenation.
 - If you are using the ACC1IDxx statement as described in “Updating the GPZPUMP startup procedure” on page 32, add the following statement:
`//ACC1IDxx DD DUMMY`

- Optional: Customize the messages and exceptions that are processed by Event Pump. Edit the AF/OPERATOR client-specific REXX program and include the GTMEDIAF calls. For information about the GTMEDIAF program, see "Customizing AF/OPERATOR to send events."
- Save the REXX code in a library that is available to the SYSEXEC DD statement of the AF/OPERATOR started task. If the library consists of a concatenation of several data sets, save the REXX code to the first data set.

Reviewing the Event Pump GPZPARMS member

Make sure that the following parameters are specified in the GPZPARMS member:

- AFOPER_JOBNAME=*stcname*[,*stcname*[...]]
The *stcname*= started task name for the AF/OPERATOR task.
- EDI_BUFFER_SIZE=5000
A size value of 5000 is required for the AF/OPERATOR interface implementation.

Usage note: In a 3-address-space environment, specify the startup parameters in the PARMPUMP member. The source collector startup procedure is GTMPUMP.

Customizing AF/OPERATOR to send events

Event Pump provides an EDI interface for the AF/OPERATOR event feed. AF/OPERATOR provides status information about the monitored components of the enterprise. Messages and exceptions are processed by Event Pump and sent to the EIF probe.

The GTMEDITB EDI table member contains all format, action, and data types that are supported by Event Pump. The GTMEDITB member is used by the EDI to verify the data that is sent. The GTMEDIAF REXX program runs within the AF/OPERATOR product environment. If the data that is sent is not valid, a WTO message is issued.

Calling the EDI

You can add commands to your automation routines to call the GTMEDIAF REXX program and to pass parameters.

AF/OPERATOR (pass MESSAGE):

```
CALL "GTMEDIAF" START_OF_DATA
Token = "GTM" || result
CALL "GTMEDIAF" TOKEN 10 02          /* AFOPERATOR MESSAGE EVENT */
CALL "GTMEDIAF" TOKEN 05 IP01        /* SMF ID */
CALL "GTMEDIAF" TOKEN 03 CICSREG1    /* OBJECT NAME */
CALL "GTMEDIAF" TOKEN 08 IEF403I     /* MESSAGE ID */
CALL "GTMEDIAF" TOKEN 49 'JOB FTP STARTED' /* MESSAGE TEXT */
CALL "GTMEDIAF" TOKEN END_OF_DATA TRACEOFF
```

Starting and ending the automation process

Sending notifications that your automation process is starting or ending is similar to the process of sending messages. The automation routine passes similar information to the REXX routine in the message process.

When automation begins, the EDI receives the following message:

Automation starting for object xxxxxxxx

When automation ends, the following message is sent to the EDI:

Automation ending for object xxxxxxxx

xxxxxxx

Indicates a resource name (for example, DFHSM or HSC).

The following example notifies Event Pump that automation is in progress for a specific resource:

```
CALL "GTEDIAF" START_OF_DATA
Token = "GTM" || rc
CALL "GTEDIAF" TOKEN 10 02      /* AFOperator MESSAGE EVENT */
CALL "GTEDIAF" TOKEN 05 IP01    /* SMF ID */
CALL "GTEDIAF" TOKEN 03 BATCH001 /* OBJECT NAME */
CALL "GTEDIAF" TOKEN 08 GTMAUON /* MESSAGE ID indicating automation on */
CALL "GTEDIAF" TOKEN 49 "Automation starting for object BATCH001" /* MESSAGE TEXT */
CALL "GTEDIAF" TOKEN END_OF_DATA TRACEOFF
```

The following example notifies Event Pump that the automation procedure completed for a specific resource:

```
CALL "GTEDIAF" START_OF_DATA
Token = "GTM" || rc
CALL "GTEDIAF" TOKEN 10 02      /* AFOperator MESSAGE EVENT */
CALL "GTEDIAF" TOKEN 05 IP01    /* SMF ID */
CALL "GTEDIAF" TOKEN 03 BATCH001 /* OBJECT NAME */
CALL "GTEDIAF" TOKEN 08 GTMAUOF /* MESSAGE ID indicating automation off */
CALL "GTEDIAF" TOKEN 49 "Automation ending for object BATCH001" /* MESSAGE TEXT */
CALL "GTEDIAF" TOKEN END_OF_DATA TRACEOFF
```

The following example notifies Event Pump that the automation procedure ended for a specific resource:

```
CALL "GTEDIAF" START_OF_DATA
Token = "GTM" || rc
CALL "GTEDIAF" TOKEN 10 02      /* AFOperator MESSAGE EVENT */
CALL "GTEDIAF" TOKEN 05 IP01    /* SMF ID */
CALL "GTEDIAF" TOKEN 03 BATCH001 /* OBJECT NAME */
CALL "GTEDIAF" TOKEN 08 GTMAUAB /* MESSAGE ID indicating automation off */
CALL "GTEDIAF" TOKEN 49 "Automation terminating for object BATCH001" /* MESSAGE TEXT */
CALL "GTEDIAF" TOKEN END_OF_DATA TRACEOFF
```

Additional modification examples

The following examples show additional requests that you can make with AF/OPERATOR.

```
/* Pass MESSAGE to Object */
CALL "GTEDIAF" START_OF_DATA
Token = "GTM" || result
CALL "GTEDIAF" TOKEN 10 02      /* AF/OPER MESSAGE EVENT */
CALL "GTEDIAF" TOKEN 05 IP01    /* SMF ID */
CALL "GTEDIAF" TOKEN 03 CICSREG1 /* OBJECT NAME */
CALL "GTEDIAF" TOKEN 08 SAM001I /* MESSAGE ID */
CALL "GTEDIAF" TOKEN 49 'Test message from AF Operator'/*MESSAGE*/
CALL "GTEDIAF" TOKEN END_OF_DATA TRACEOFF
```

```
/* Pass EXCEPTION to Object */
CALL "GTEDIAF" START_OF_DATA
Token = "GTM" || result
CALL "GTEDIAF" TOKEN 10 03      /* AF/OPERATOR EXCEPTION EVENT */
CALL "GTEDIAF" TOKEN 05 IP01    /* SMF ID */
CALL "GTEDIAF" TOKEN 03 CICSREG2 /* OBJECT NAME */
CALL "GTEDIAF" TOKEN 08 TMAX    /* EXCEPTION */
CALL "GTEDIAF" TOKEN 49 500     /* EXCEPTION VALUE */
CALL "GTEDIAF" TOKEN END_OF_DATA TRACEOFF
```

```
/* Pass RESOLUTION to Object */
CALL "GTEDIAF" START_OF_DATA
Token = "GTM" || result
```

```
CALL "GTMEDIAF" TOKEN 10 03          /* AF/OPERATOR EXCEPTION EVENT */
CALL "GTMEDIAF" TOKEN 05 IPO1        /* SMF ID */
CALL "GTMEDIAF" TOKEN 03 CICSREG2    /* OBJECT NAME */
CALL "GTMEDIAF" TOKEN 08 TMAX        /* EXCEPTION */
CALL "GTMEDIAF" TOKEN 49 'OK'        /* EXCEPTION VALUE */
CALL "GTMEDIAF" TOKEN END_OF_DATA TRACEOFF
```

Error handling

When the EDI cannot send data to Event Pump, the EDI gets a return code of greater than 0 and a GTMxx101 message is sent to Netcool/OMNIBus. When the EDI failure is resolved, a GTMxx102 message is sent to Netcool/OMNIBus. See “API exit routine return codes” on page 176 for a list of return codes.

Modifying the source

To support the passing of data from a source product or application to Event Pump through the EDI, make the following modifications to the application capture within the originating external product or application.

Pass the data in the following order:

1. START_OF_DATA
2. Format type and action type
3. Data type and data values
4. END_OF_DATA

See “Tivoli AF/OPERATOR (format type 10)” on page 180 for information about specific format types and action types.

The following example describes how to pass a state change using the EDI:

```
CALL "GTMEDIAF" START_OF_DATA
Token = "GTM" || rc
CALL "GTMEDIAF" TOKEN 10 01          /* AFOPERATOR STATE CHANGE EVENT */
CALL "GTMEDIAF" TOKEN 05 IPO1        /* SMF ID */
CALL "GTMEDIAF" TOKEN 06 SUBSYSTEM   /* RESOURCE TYPE */
CALL "GTMEDIAF" TOKEN 03 STCTEST1    /* OBJECT NAME */
CALL "GTMEDIAF" TOKEN 08 ABENDING    /* STATE CHANGE */
CALL "GTMEDIAF" TOKEN END_OF_DATA TRACEOFF
```

Example notes:

1. The TOKEN unique value is used to distinguish between concurrent calls to the EDI. With the use of the TOKEN value, data is not lost between calls to the EDI.
2. The first parameter that is passed is START_OF_DATA. This parameter initializes the EDI to start accepting information.
3. The second parameter that is passed contains the format type and action type. In this case, the format type is 10 and the action type is 01. You can pass the format type with or without the action type in numeric or character format.

The second parameter can also be one of the following format types:

- "GTMEDIAF" 10 STATE_CHANGE
- "GTMEDIAF" AFOPERATOR STATE_CHANGE
- "GTMEDIAF" AFOPERATOR 01

4. The next parameters that are passed contain data type and data:

Table 7. Parameters that contain data type and data

Data type	Data
05 (SMFID)	IPO1 (name of MVS image)
06 (RESOURCE_TYPE)	SUBSYSTEM

Table 7. Parameters that contain data type and data (continued)

Data type	Data
03 (OBJECT_NAME)	STCTEST1
08 (EVENT_TYPE)	ABENDING

5. END_OF_DATA and TRACEOFF are the last parameters passed. END_OF_DATA notifies EDI that there are no more records to pass. TRACEOFF informs the EDI not to print out the record that is sent to Event Pump. If TRACEON is passed, the record that is passed to Event Pump is printed. This record is useful for troubleshooting a problem with the EDI.

Usage notes:

1. Format, action, and data type are passed in their numeric or character formats.
2. Use single quotation marks (') when passing text, for example, message text. The following example posts the Test message from AF/OPERATOR message with a message ID of SAM001I to the BATCH001 resource:

```
CALL "GTEDIAF" START_OF_DATA
Token = "GTM" || rc
CALL "GTEDIAF" TOKEN 10 02          /* AFOPERATOR MESSAGE EVENT */
CALL "GTEDIAF" TOKEN 05 IPO1        /* SMF ID */
CALL "GTEDIAF" TOKEN 03 BATCH001    /* OBJECT NAME */
CALL "GTEDIAF" TOKEN 08 SAM001I     /* MESSAGE ID */
CALL "GTEDIAF" TOKEN 49 "Test message from AFOPERATOR" /* MESSAGE TEXT */
CALL "GTEDIAF" TOKEN END_OF_DATA TRACEOFF
```

3. An optional TRACEPROG parameter is also available to trace the entire EDI program (GTEDIAF). You can trace all call statements or specific ones.

An example of tracing all call statements follows:

```
CALL "GTEDIAF" TRACEPROG 'START_OF_DATA
Token = "GTM" || rc
CALL "GTEDIAF" TRACEPROG TOKEN 10 02          /*AFOPERATOR MESSAGE EVENT */
CALL "GTEDIAF" TRACEPROG TOKEN 05 IPO1        /*SMF ID */
CALL "GTEDIAF" TRACEPROG TOKEN 03 BATCH001 /*OBJECT NAME */
CALL "GTEDIAF" TRACEPROG TOKEN 08 SAM001I /*MESSAGE ID */
CALL "GTEDIAF" TRACEPROG TOKEN 49 "Test message from AFOPERATOR " /*MESSAGE TEXT*/
CALL "GTEDIAF" TRACEPROG TOKEN END_OF_DATA TRACEOFF
```

An example of tracing specific call statements follows:

```
CALL "GTEDIAF" 'START_OF_DATA
Token = "GTM" || rc
CALL "GTEDIAF" TRACEPROG TOKEN 10 02          /*AFOPERATOR MESSAGE EVENT */
CALL "GTEDIAF" TOKEN 05 IPO1                  /*SMF ID */
CALL "GTEDIAF" TRACEPROG TOKEN 03 BATCH001 /*OBJECT NAME */
CALL "GTEDIAF" TOKEN 08 SAM001I              /*MESSAGE ID */
CALL "GTEDIAF" TOKEN 49 "Test message from AFOPERATOR " /*MESSAGE TEXT */
CALL "GTEDIAF" TOKEN END_OF_DATA TRACEOFF
```

Tivoli NetView for z/OS automation

The Event Pump interfaces directly with automation that is based on the NetView for z/OS program through the implementation of the External Data Interface (EDI). The EDI enables an external application program to pass text-based messages directly to Event Pump on the same z/OS system.

To enable Event Pump to track messages or exceptions that are passed from console-written NetView for z/OS REXX programs, follow these steps:

1. Edit the NetView for z/OS specific REXX program and include the necessary GTMEDISA program calls.

2. Update the NetView for z/OS startup procedure (CNMPROC) as follows:
 - After you modify the REXX program, save it in a data set that is specified on the DSICLD DD statement. It is preferable to use the first data set in the concatenation.
 - Add the *hlq*.SGTMEXEC library to the DSICLD concatenation.
 - Add the *hlq*.SGTMMODS library to the STEPLIB concatenation.
 - If you added the ACC1IDxx statement to the GPZPUMP procedure, add the following statement:

```
//ACC1IDxx DD DUMMY
```

where *xx* is a 2-character suffix that defines the instance.
3. Make sure that the following parameters are specified in the GPZPARMS member:
 - SA390_JOBNAME=*stcname*[,*stcname*[,...]]
 The *stcname*= started task name for the SA390 task.
 - EDI_BUFFER_SIZE=5000
 A size value of 5000 is required for the SA390 interface implementation.

Usage note: In a 3-address-space environment, specify the startup parameters in the PARMUMP member. The source collector startup procedure is GTMPUMP.

Tivoli System Automation for z/OS

These components enable the integration of Tivoli System Automation for z/OS with Event Pump:

- System Automation for z/OS application programming interface (API), which is available through the general status observer facility and implemented in the automation manager.
 You can register an exit to gather status information for specific resources or resource types. Event Pump uses the exit to capture all application (APL) and application group (APG) information.
- Tivoli NetView for z/OS program-to-program interface (PPI), which passes information from the general status observer facility exit to Event Pump.
 The PPI is the primary interface for this implementation.
- Event Pump EDI interface.
- The source collector passes data to the EIF probe for processing.

Table 8 provides an overview of the tasks required to configure System Automation for z/OS to interface with Event Pump.

Table 8. Implementing the System Automation for z/OS event feed

Task	Where to find information
Install the System Automation for z/OS REXX interface.	"Installing System Automation for z/OS" on page 59
Define the IHSSAUT1 and IHSSAUT2 automation operators.	"Updating the System Automation for z/OS control file" on page 59
Review the customization of the NetView program.	"Customizing the NetView program to enable Event Pump to process events" on page 60

Table 8. Implementing the System Automation for z/OS event feed (continued)

Task	Where to find information
Specify resources to be monitored by updating the IHSSMONI sample.	"Specifying the resources to be monitored" on page 60
Customize System Automation to send events.	"Customizing System Automation for z/OS to send events" on page 60
Restart the NetView program.	"Restarting the NetView program" on page 64
Verify the installation.	"Verifying the installation" on page 64

Event flow

When the status of a System Automation for z/OS application or group changes, the following event flow occurs:

1. The System Automation for z/OS status observer passes the status information to the Event Pump IHSSOBSX exit, which is registered with the System Automation for z/OS status observer.
2. The Event Pump IHSSOBSX exit passes the relevant data to Event Pump using the NetView program-to-program interface.
3. Event Pump passes the data to the data space, which places the data on the event distributor queue.
4. Event Pump forwards the event to the EIF probe.

The following scenario is an event scenario for System Automation for z/OS:

1. The APPC started task that is monitored by System Automation for z/OS moves from a satisfactory to problem state.
2. The System Automation for z/OS status observer calls the Event Pump IHSSOBSX exit and passes the System Automation for z/OS resource name and the System Automation for z/OS status types (Observed, Desired, Automation, Startability, and Compound).
3. Event Pump IHSSOBSX exit checks the monitoring level and previous status of the APPC task, detects that the status has changed, and calls the IHSMMSG program to send the status change to Event Pump that is using the NetView program-to-program interface. The record that is sent also contains the resource type (APL or APG) and the MVS job name.
4. Event Pump passes the data to the data space, which places the data on the event distributor queue.
5. Event Pump forwards the event to the EIF probe.

Installing System Automation for z/OS

Verify that the System Automation for z/OS REXX executable files, samples, messages, and sample members are installed. These data sets are installed as part of the SMP/E installation of the product. If needed, the installation instructions can be found in the program directory.

Updating the System Automation for z/OS control file

Define the IHSSAUT1 and IHSSAUT2 operator IDs in your automation control file using the customization dialog boxes:

1. From the Entry Type Selection list, select option **37** and press Enter.

2. From the Entry Name Selection list, type **NEW** on the command line and press Enter.
3. From the Define New Entry list, type TBSM_AUTO_OPS on the Name line and Event Pump Autotasks on the Short Description line. Press F3.
4. From the Policy Selection list, select **OPERATORS** and press Enter.
5. From the Automation Operator Definitions list, define the IHSSAUT1 and IHSSAUT2 operators. Press Enter.
6. From the Automation Operator NetView Userids list, type IHSSAUT1 as the primary operator and press F3. Repeat this step for the IHSSAUT2 operator.
7. Update the **Where Used** field.
8. Rebuild your automation control file and issue the INGAMS command from the NetView command line on your System Automation for z/OS agent.

Customizing the NetView program to enable Event Pump to process events

Ensure that all the steps in Chapter 21, “Tivoli NetView for z/OS,” on page 123 are completed.

When the AOF540I message is trapped in the NetView message automation table, the Event Pump to System Automation for z/OS interface is initialized, which runs the IHSSSTRT program. Use the following steps to initialize the IHSSSTRT program:

1. Check that all Event Pump and System Automation for z/OS autotasks are active. If they are not, activate them.
2. Clear the in-store cache.
3. Check the value of the DSIPARM(IHS\$PARM) parameters and create the associated global variables.
4. Register the IHSSOBSX exit with the System Automation for z/OS status observer.
5. Process the IHSSMONI sample (see “Specifying the resources to be monitored”).

Issue the following command from the NetView command line to manually begin initialization:

```
EXCMD IHSSAUT1,IHSSSTRT
```

Specifying the resources to be monitored

You can control the System Automation for z/OS status type for resources that are forwarded to Netcool/OMNIBus. Customize the IHSSMONI sample that is located in the NetView DSIPARM DD concatenation. For information on how to customize the IHSSMONI sample, see the comments in the sample.

If the IHSSMONI sample is modified while the System Automation for z/OS automation agent is running, incorporate the changes by issuing the IHSSSTRT command from the NetView command line.

Customizing System Automation for z/OS to send events

Event Pump provides an EDI interface for IBM Tivoli System Automation for z/OS. The System Automation for z/OS product is used to monitor the status and condition of enterprise components. Event Pump uses message and exceptions that are received by the NetView for z/OS product.

The GTMEDITB EDI table member contains all format, action, and data types that are supported by Event Pump. The GTMEDISA REXX program runs within the System Automation product environment. The GTMEDITB member is used by the EDI to verify the data that is sent. If the data sent is not valid, a WTO message is issued.

Calling the EDI

You can add commands to your automation routines to call the GTMEDISA REXX program and to pass parameters. The syntax of the program call is dependent on the event feed.

System Automation for z/OS (pass STATE_CHANGE):

```
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 01          /* SA390 STATE CHANGE EVENT */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 06 SUBSYSTEM   /* RESOURCE TYPE */
'GTMEDISA' TOKEN 03 TESTSTC1    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 ABENDING    /* STATE CHANGE */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF
```

Starting and ending the automation process

Sending notifications that your automation process is starting or ending is similar to the process of sending messages. The automation routine passes similar information to the REXX routine in the message process.

When automation begins, the EDI receives the following message:

Automation starting for object xxxxxxxx

When automation ends, the following message is sent to the EDI:

Automation ending for object xxxxxxxx

xxxxxxx

Indicates a resource name (for example DFHSM or HSC).

The following example notifies Event Pump that automation is in progress for a specific resource:

```
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 02          /* SA390 MESSAGE EVENT */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 03 BATCH001    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 GTMAUON     /* MESSAGE ID indicating automation on */
'GTMEDISA' TOKEN 49 "Automation starting for object BATCH001" /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF
```

The following example notifies Event Pump that the automation procedure completed for a specific resource:

```
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 02          /* SA390 MESSAGE EVENT */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 03 BATCH001    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 GTMAUOF     /* MESSAGE ID indicating automation off */
'GTMEDISA' TOKEN 49 "Automation ending for object BATCH001" /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF
```

The following example notifies Event Pump that the automation procedure ended for a specific resource:

```

'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 02          /* SA390 MESSAGE EVENT */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 03 BATCH001    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 GTMAUAB     /* MESSAGE ID indicating automation off */
'GTMEDISA' TOKEN 49 "Automation terminating for object BATCH001" /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF

```

Additional examples of requests

The following examples show additional requests that you can make with the System Automation for z/OS event feed.

System Automation for z/OS

```

/* Pass MESSAGE to Object */
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 02          /* SA390 MESSAGE EVENT */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 03 CICSREG1    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 SAM001I     /* MESSAGE ID */
'GTMEDISA' TOKEN 49 "Test message from NetView" /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF

/* Pass EXCEPTION to Object */
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 03          /* SA390 EXCEPTION EVENT */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 03 CICSREG2    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 TMAX        /* EXCEPTION */
'GTMEDISA' TOKEN 49 500         /* EXCEPTION VALUE */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF

/* Pass IMS data */
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 18 03          /* IMS MESSAGE EVENT */
'GTMEDISA' TOKEN 02 IMSS        /* IMS OBJECT TYPE */
'GTMEDISA' TOKEN 15 USIBMNT.NTA7MVS /*NETWORK NAME AND VTAM APPL ID */
'GTMEDISA' TOKEN 16 IP01        /* SMF ID */
'GTMEDISA' TOKEN 17 IMA1CTL      /* IMS SUBSYSTEM NAME */
'GTMEDISA' TOKEN 63 THRESHOLD    /* STATE OF OBJECT */
'GTMEDISA' TOKEN 64 RM0710W      /* MESSAGE ID */
'GTMEDISA' TOKEN 65 "Long message queue > x % ..." /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF

/* Pass OPC Exception data */
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 05          /* OPC Exception EVENT */
'GTMEDISA' TOKEN 03 OPCTST01     /* OBJECT NAME */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 08 'BATCHREPHR' /* EXCEPTION */
'GTMEDISA' TOKEN 10 OPCTST01     /* OPC APPLICATION NAME */
'GTMEDISA' TOKEN 11 000406005    /* OPC INPUT ARRIVAL TIME */
'GTMEDISA' TOKEN 12 CPU1        /* OPC WORKSTATION ID */
'GTMEDISA' TOKEN 49 'REPLY FOR JOB OPCTST01' /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF

```

Error handling

When the EDI cannot send data to Event Pump, the EDI gets a return code of greater than 0 and a GTMxx101 message is sent to Netcool/OMNIbus. When the EDI failure is resolved, a GTMxx102 message is sent to Netcool/OMNIbus. See “API exit routine return codes” on page 176 for a list of return codes.

Modifying the source

To support the passing of data from a source product or application to Event Pump through the EDI, make the following modifications to the application capture within the originating external product or application.

Pass the data in the following order:

1. START_OF_DATA
2. Format type and action type
3. Data type and data values
4. END_OF_DATA

See “Tivoli System Automation for z/OS (format type 07)” on page 181 for information about specific format types and action types.

The following example describes how to pass a state change using the System Automation for z/OS EDI:

```
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 01          /* SA390 STATE CHANGE EVENT */
'GTMEDISA' TOKEN 05 IPO1        /* SMF ID */
'GTMEDISA' TOKEN 06 SUBSYSTEM   /* RESOURCE TYPE */
'GTMEDISA' TOKEN 03 STCTEST1    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 ABENDING    /* STATE CHANGE */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF
```

Example notes:

1. The TOKEN unique value is used to distinguish between concurrent calls to the EDI. With the use of the TOKEN value, data is not lost between calls to the EDI.
2. The first parameter that is passed is START_OF_DATA. This parameter initializes the EDI to start accepting information.
3. The second parameter that is passed contains the format type and action type. In this case, the format type is 07 and the action type is 01. You can pass the format type with or without the action type in numeric or character format.
The second parameter can also be one of the following format types:
 - 'GTMEDISA' 07 STATE_CHANGE
 - 'GTMEDISA' SA390 STATE_CHANGE
 - 'GTMEDISA' SA390 01
4. The next parameters that are passed contain data type and data:

Table 9. Parameters that contain data type and data

Data Type	Data
05 (SMFID)	IPO1 (name of MVS image)
06 (RESOURCE_TYPE)	SUBSYSTEM
03 (OBJECT_NAME)	STCTEST1
08 (EVENT_TYPE)	ABENDING

5. END_OF_DATA and TRACEOFF are the last parameters that are passed. END_OF_DATA notifies the EDI that there are no more records to pass. TRACEOFF informs the EDI not to print out the record that is sent to Event Pump. If TRACEON is passed, the record passed to Event Pump is printed. This record is useful for troubleshooting a problem with the EDI.

Usage notes:

1. Format, action, and data type are passed in their numeric or character formats.

2. Use single quotation marks (') when passing text, for example, message text. The following example posts the Test message from SA390 message with a message ID of SAM001I to the BATCH001 resource:

```
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TOKEN 07 02          /* SA390 MESSAGE EVENT */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 03 BATCH001    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 SAM001I     /* MESSAGE ID */
'GTMEDISA' TOKEN 49 "Test message from SA390" /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF
```

3. An optional TRACEPROG parameter is also available to trace the entire EDI program (GTMEDISA). You can trace all call statements or specific ones.

An example of tracing all call statements follows:

```
'GTMEDISA' TRACEPROG START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TRACEPROG TOKEN 07 02          /*SA390 MESSAGE EVENT */
'GTMEDISA' TRACEPROG TOKEN 05 IP01        /*SMF ID */
'GTMEDISA' TRACEPROG TOKEN 03 BATCH001    /*OBJECT NAME */
'GTMEDISA' TRACEPROG TOKEN 08 SAM001I     /*MESSAGE ID */
'GTMEDISA' TRACEPROG TOKEN 49 "Test message from SA390 " /*MESSAGE TEXT*/
'GTMEDISA' TRACEPROG TOKEN END_OF_DATA TRACEOFF
```

An example of tracing specific call statements follows:

```
'GTMEDISA' START_OF_DATA
Token = "GTM" || rc
'GTMEDISA' TRACEPROG TOKEN 07 02          /*SA390 MESSAGE EVENT */
'GTMEDISA' TOKEN 05 IP01                  /*SMF ID */
'GTMEDISA' TRACEPROG TOKEN 03 BATCH001    /*OBJECT NAME */
'GTMEDISA' TOKEN 08 SAM001I               /*MESSAGE ID */
'GTMEDISA' TOKEN 49 "Test message from SA390 " /*MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA TRACEOFF
```

Restarting the NetView program

When you complete all customization steps, restart the NetView program. If this restart is the first time that the NetView program is started with the installed Event Pump program, it automatically detects all System Automation for z/OS resources for that system.

Verifying the installation

Verify the installation with the following steps:

- “Verify host customization”
- “Verify that the Event Pump-System Automation for z/OS interface is started”
- “Verify System Automation for z/OS event logging” on page 65

Verify host customization

You can test the configuration steps for the host code by using the Installation Verification Procedures (IVP) program. To run the IVP program, issue the following command to the NetView program, and correct any errors that are reported:

```
IHS$DEBUG IVP SA
```

Verify that the Event Pump-System Automation for z/OS interface is started

View the current NetView network log on the System Automation for z/OS agent and check for the AOF540 message. Event Pump-System Automation for z/OS is started using a NetView message trap for this message. When initialization is complete, the IHS708I message is issued. Check that this message is sent to your NetView network log.

Verify System Automation for z/OS event logging

Recycle a sample System Automation for z/OS managed resource using System Automation for z/OS, for example, the Resource Measurement Facility (RMF). Check the NetView network log for the following records:

```
IHS720I IHS$EVTQ 07\01 02=APL|03=RMF|
```

Disable logging the events that are sent to the PPI receiver

The IHS_LOGGING parameter in the IHS\$PRM member is set to ON as the default value. This results in all Event Pump events being logged to the NetView network log.

When installation is complete and verified, reset this parameter to OFF. Create an IHS_LOGGING parameter in the IHS\$UPRM member with a value of OFF.

BMC MainView AutoOPERATOR for z/OS

Event Pump interfaces with the BMC MainView AutoOPERATOR for z/OS automation product. With the implementation of the EDI, an external application program can pass text-based messages directly to Event Pump on the same z/OS system.

Table 10 provides an overview of the tasks that are required to configure AutoOPERATOR to interface with Event Pump.

Table 10. Implementing the BMC MainView AutoOPERATOR event feed

Task	Where to find information
Install the AutoOPERATOR EDI interface.	"Installing the AutoOPERATOR EDI interface"
Review the GPZPARMS member and update as necessary.	"Reviewing the Event Pump GPZPARMS member" on page 66
Customize the resource monitoring levels for AutoOPERATOR resources.	"Customizing BMC MainView AutoOPERATOR to send events" on page 66

Installing the AutoOPERATOR EDI interface

Make the following modifications to install the AutoOPERATOR EDI interface:

1. Edit the AutoOPERATOR started task and make the following updates:
 - Add the `hlq.SGTMEEXEC` data set to the SYSEXEC DD concatenation.
 - Add the `hlq.SGTMMODS` data set to the STEPLIB DD concatenation.
 - If you are using the ACC1IDxx statement as described in "Updating the GPZPUMP startup procedure" on page 32, add the following statement:
`//ACC1IDxx DD DUMMY`
2. Optional: Customize the messages and exceptions that are processed by Event Pump. Edit the AutoOPERATOR client-specific REXX program and include the GTMEDIAO calls. For information about the GTMEDIAO program, see "Customizing BMC MainView AutoOPERATOR to send events" on page 66.
3. Save the REXX code in a library that is available to the SYSEXEC DD statement of the AutoOPERATOR started tasks. If the library consists of a concatenation of several data sets, save the REXX code to the first data set.

Reviewing the Event Pump GPZPARMS member

Make sure that the following parameters are specified in the GPZPARMS member:

- AUTOOPER_JOBNAME=*stcname*[,*stcname*[,...]]

The *stcname*= started task name for the AutoOPERATOR task.

- EDI_BUFFER_SIZE=5000

A size value of 5000 is required for the AutoOPERATOR interface implementation.

Usage note: In a 3-address-space environment, specify the startup parameters in the PARMPUMP member. The source collector startup procedure is GTMPUMP.

Customizing BMC MainView AutoOPERATOR to send events

Event Pump provides an EDI interface for the AutoOPERATOR event feed. The BMC MainView AutoOPERATOR product provides status information about the monitored components of the enterprise. Messages and exceptions are processed by Netcool/OMNIBus Server and their affect is recorded on the component resource within the Tivoli Business Service Manager representation of the enterprise.

The GTMEDITB EDI table member contains all format, action, and data types that are supported by Event Pump. The GTMEDITB member is used by the EDI interface to verify the data that is sent. The GTMEDIAO REXX program runs within the AutoOPERATOR product environment. If the data that is sent is not valid, a Write to Operator (WTO) message is issued.

Calling the EDI

You can add commands to your automation routines to call the GTMEDIAO REXX program and to pass parameters.

AutoOPERATOR (pass MESSAGE):

```
CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || result
CALL "GTMEDIAO" TOKEN 12 02          /* AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01        /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 CICSREG1    /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 IEF403I     /* MESSAGE ID */
CALL "GTMEDIAO" TOKEN 49 "JOB FTP STARTED" /* MESSAGE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF
```

Starting and ending the automation process

Sending notifications that your automation process is starting or ending is similar to the process of sending messages. The automation routine passes similar information to the REXX routine in the message process.

When automation begins, the EDI interface receives the following message:

Automation starting for object xxxxxxxx

When automation ends, the following message is sent to the EDI interface:

Automation ending for object xxxxxxxx

xxxxxxx

Indicates a resource name (for example DFHSM or HSC).

The following example notifies Event Pump that automation is in progress for a specific resource:

```
CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || rc
CALL "GTMEDIAO" TOKEN 12 02          /* AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01        /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 BATCH001    /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 GTMAUON     /* MESSAGE ID indicating automation on */
CALL "GTMEDIAO" TOKEN 49 "Automation starting for object BATCH001" /* MESSAGE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF
```

The following example notifies Event Pump that the automation procedure is completed for a specific resource:

```
CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || rc
CALL "GTMEDIAO" TOKEN 12 02          /* AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01        /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 BATCH001    /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 GTMAUOF     /* MESSAGE ID indicating automation off */
CALL "GTMEDIAO" TOKEN 49 "Automation ending for object BATCH001" /* MESSAGE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF
```

The following example notifies Event Pump that the automation procedure ended for a specific resource:

```
LL "GTMEDIAO" START_OF_DATA
Token = "GTM" || rc
CALL "GTMEDIAO" TOKEN 12 02          /* AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01        /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 BATCH001    /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 GTMAUAB     /* MESSAGE ID indicating automation off */
CALL "GTMEDIAO" TOKEN 49 "Automation terminating for object BATCH001" /* MESSAGE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF
```

Additional modification examples

The following examples show additional requests that you can make with the AutoOPERATOR event feed.

AutoOPERATOR

```
/* Pass MESSAGE to Object */
CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || result
CALL "GTMEDIAO" TOKEN 12 02          /* AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01        /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 CICSREG1    /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 SAM001I     /* MESSAGE ID */
CALL "GTMEDIAO" TOKEN 49 "Test message from AutoOperator" /* MESSAGE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF

/* Pass MESSAGE to Object */
CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || result
CALL "GTMEDIAO" TOKEN 12 02          /* AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01        /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 CICSREG1    /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 IEF404I     /* MESSAGE ID */
CALL "GTMEDIAO" TOKEN 49 "JOB FTP ENDED" /* MESSAGE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF

/* Pass EXCEPTION to Object */
CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || result
CALL "GTMEDIAO" TOKEN 12 03          /* AutoOPERATOR EXCEPTION EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01        /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 CICSREG2    /* OBJECT NAME */
```

```

CALL "GTMEDIAO" TOKEN 08 TMAX          /* EXCEPTION */
CALL "GTMEDIAO" TOKEN 49 500          /* EXCEPTION VALUE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF

/* Pass RESOLUTION to Object */
CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || result
CALL "GTMEDIAO" TOKEN 12 03           /* AutoOPERATOR EXCEPTION EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01         /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 CICSREG2     /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 TMAX         /* EXCEPTION */
CALL "GTMEDIAO" TOKEN 49 'OK'         /* EXCEPTION VALUE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF

```

Error handling

When the EDI cannot send data to Event Pump, the EDI interface gets a non-zero return code and a GTMAO101 message is printed to the system log. When the EDI failure is resolved, a GTMAO102 message is printed to the system log. See “API exit routine return codes” on page 176 for a list of return codes.

Modifying the source

To support the passing of data from a source product or application to Event Pump through the EDI interface, make the following modifications to the application capture within the originating external product or application.

Pass the data in the following order:

1. START_OF_DATA
2. Format type and action type
3. Data type and data values
4. END_OF_DATA

See “BMC MainView AutoOPERATOR for z/OS (format type 12)” on page 177 for information about specific format types and action types.

The following example describes how to pass a state change using the EDI interface:

```

CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || rc
CALL "GTMEDIAO" TOKEN 12 01           /* AutoOPERATOR STATE CHANGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IP01         /* SMF ID */
CALL "GTMEDIAO" TOKEN 06 SUBSYSTEM    /* RESOURCE TYPE */
CALL "GTMEDIAO" TOKEN 03 STCTEST1     /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 ABENDING     /* STATE CHANGE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF

```

Example notes:

1. The TOKEN unique value is used to distinguish between concurrent calls to the EDI interface. With the use of the TOKEN value, data is not lost between calls to the EDI interface.
2. The first parameter that is passed is START_OF_DATA. This parameter initializes the EDI interface to start accepting information.
3. The second parameter that is passed contains the format type and action type. In this case, the format type is 12 and the action type is 01. You can pass the format type with or without the action type in numeric or character format.

The second parameter can also be one of the following format types:

- "GTMEDIAO" 12 STATE_CHANGE
- "GTMEDIAO" AUTOOPERATOR STATE_CHANGE
- "GTMEDIAO" AUTOOPERATOR 01

4. The next parameters that are passed contain data type and data:

Table 11. Parameters that contain data type and data

Data Type	Data
05 (SMFID)	IPO1 (name of MVS image)
06 (OBJECT_TYPE)	SUBSYSTEM
03 (OBJECT_NAME)	STCTEST1
08 (STATE_MSGID_EXCPID)	ABENDING

5. END_OF_DATA and TRACEOFF are the last parameters passed. END_OF_DATA notifies the EDI interface that there are no more records to pass. TRACEOFF informs the EDI interface not to print out the record that is sent to Event Pump. If TRACEON is passed, the record that is passed to Event Pump is printed. This record is useful for troubleshooting a problem with the EDI interface.

Usage notes:

1. Format, action, and data type are passed in their numeric or character formats.
2. Use single quotation marks (') and double quotation marks (") when passing text, for example, message text. The following example posts the Test message from AutoOPERATOR message with a message ID of SAM001I to the BATCH001 resource:

```
CALL "GTMEDIAO" START_OF_DATA
Token = "GTM" || rc
CALL "GTMEDIAO" TOKEN 12 02      /* AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IPO1    /* SMF ID */
CALL "GTMEDIAO" TOKEN 03 BATCH001 /* OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 SAM001I /* MESSAGE ID */
CALL "GTMEDIAO" TOKEN 49 "Test message from AutoOPERATOR" /* MESSAGE */
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF
```

3. An optional TRACEPROG parameter is also available to trace the entire EDI program (GTMEDIAO). You can trace all call statements or specific ones.

An example of tracing all call statements follows:

```
CALL "GTMEDIAO" TRACEPROG START_OF_DATA
Token ="GTM"||rc
CALL "GTMEDIAO" TRACEPROG TOKEN 12 02      /*AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TRACEPROG TOKEN 05 IPO1    /*SMF ID */
CALL "GTMEDIAO" TRACEPROG TOKEN 03 BATCH001 /*OBJECT NAME */
CALL "GTMEDIAO" TRACEPROG TOKEN 08 SAM001I /*MESSAGE ID */
CALL "GTMEDIAO" TRACEPROG TOKEN 49 "Test message from AutoOPERATOR " /*MESSAGE*/
CALL "GTMEDIAO" TRACEPROG TOKEN END_OF_DATA TRACEOFF
```

An example of tracing specific call statements follows:

```
CALL "GTMEDIAO" START_OF_DATA
Token ="GTM"||rc
CALL "GTMEDIAO" TRACEPROG TOKEN 12 02      /*AutoOPERATOR MESSAGE EVENT */
CALL "GTMEDIAO" TOKEN 05 IPO1              /*SMF ID */
CALL "GTMEDIAO" TRACEPROG TOKEN 03 BATCH001 /*OBJECT NAME */
CALL "GTMEDIAO" TOKEN 08 SAM001I           /*MESSAGE ID */
CALL "GTMEDIAO" TOKEN 49 "Test message from AutoOPERATOR " /*MESSAGE*/
CALL "GTMEDIAO" TOKEN END_OF_DATA TRACEOFF
```

4. See "API exit routine return codes" on page 176 for return code information.

Computer Associates OPS/MVS

Event Pump interfaces with Computer Associates OPS/MVS external automation product through the EDI. The CA OPS/MVS program can pass text-based messages directly to Event Pump on the same MVS system.

Table 12 provides an overview of the tasks that are required to configure CA OPS/MVS to interface with Event Pump.

Table 12. Implementing the CA OPS/MVS event feed

Task	Where to find information
Install the CA OPS/MVS EDI interface.	"Installing the CA OPS/MVS EDI interface"
Review the GPZPARMS member and update as necessary.	"Reviewing the Event Pump GPZPARMS member"
Customize the resource monitoring levels for CA OPS/MVS resources.	"Customizing the CA OPS/MVS event feed to send events" on page 71

Installing the CA OPS/MVS EDI interface

Make the following modifications to install the CA OPS/MVS EDI interface:

1. Edit the CA OPS/MVS started task (both the server and main tasks) and make the following updates:
 - Add the *hlq*.SGTMMODS data set to the STEPLIB concatenation.
 - If you are using the ACC1IDxx statement as described in "Updating the GPZPUMP startup procedure" on page 32, add the following statement:
//ACC1IDxx DD DUMMY
2. Copy the Event Pump GTMEDIOP REXX program from the SGTMSAMP library to a library that is available within the CA OPS/MVS started tasks (both the server and main tasks) SYSEXEC DD. If applicable, update the GTMEDIOP program starting at the pipe symbol (|) to the appropriate concatenation symbol for your environment. For example, change the pipe to an exclamation point (!) if you are using the German language code page on your z/OS system. Calls to GTMEDIOP *cannot* be made from CA OPS/MVS rules (MSG, TOD, CMD). Calls can be made only from CA OPS/MVS REXX routines. These routines can then be called by CA OPS/MVS rules.
3. Optional: Customize the messages and exceptions that are processed by Event Pump. Edit the CA OPS/MVS client-specific REXX program and include the GTMEDIOP calls. For information about the GTMEDIOP program, see "Customizing the CA OPS/MVS event feed to send events" on page 71.
4. Save the REXX code in a library that is available to the SYSEXEC DD statement of the CA OPS/MVS started tasks (both the server and main tasks). It is preferable to use the first data set in the concatenation.

Reviewing the Event Pump GPZPARMS member

Make sure that the following parameters are specified in the GPZPARMS member:

- OPSMVS_JOBNAME=svrname[,svrname[,...]]

The server task name requires the server task prefix followed by an asterisk (*) to allow for all task names that are started by the server task. For example, if your server task name is OSFSRV and your main task name is CA OPSMVS, use the following code:

```
OPSMVS_JOBNAME=OPSMVS,OSF*
```

- EDI_BUFFER_SIZE=5000

A size value of 5000 is required for the CA OPS/MVS interface implementation.

Usage note: In a 3-address-space environment, specify the startup parameters that are in the PARMUMP member. The source collector startup procedure is GTMPUMP.

Customizing the CA OPS/MVS event feed to send events

Event Pump provides an EDI interface for the Computer Associates OPS/MVS event feed. The CA OPS/MVS event feed facilitates the monitoring of the enterprise and provides information about the status or condition of enterprise components. Event Pump uses the messages and exceptions that are generated through the CA OPS/MVS event feed.

The GTMEDITB EDI table member contains all format, action, and data types that are supported by Event Pump. The GTMEDITB member is used by the EDI to verify the data that is sent. The GTMEDIOP REXX program runs in the CA OPS/MVS product environment. If the data that is being sent is not valid, a WTO message is issued.

Calling the EDI

You can add commands to your automation routines to call the GTMEDIOP REXX program and to pass parameters.

CA OPS/MVS (pass STATE_CHANGE):

```
ADDRESS "TSO" "OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 01          /* OPSMVS STATE CHANGE */
"OI GTMEDIOP" TOKEN 05 IPO1        /* SMF ID */
"OI GTMEDIOP" TOKEN 06 SUBSYSTEM  /* RESOURCE TYPE */
"OI GTMEDIOP" TOKEN 03 TESTSTC1    /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 ABENDING    /* STATE CHANGE */
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF
```

Starting and ending the automation process

Sending notifications that your automation process is starting or ending is similar to the process of sending messages. The automation routine passes similar information to the REXX routine in the message process.

When automation begins, the EDI receives the following message:

Automation starting for object xxxxxxxx

When automation ends, the following message is sent to the EDI:

Automation ending for object xxxxxxxx

xxxxxxx

Indicates a resource name (for example DFHSM or HSC).

The following example notifies Event Pump that automation is in progress for a specific resource:

```
"OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 02          /* OPSMVS MESSAGE EVENT */
"OI GTMEDIOP" TOKEN 05 IPO1        /* SMF ID */
"OI GTMEDIOP" TOKEN 03 BATCH001    /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 GTMAUON     /* MESSAGE ID indicating automation on */
"OI GTMEDIOP" TOKEN 49 "Automation starting for object BATCH001" /* MESSAGE TEXT */
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF
```

The following example notifies Event Pump that the automation procedure completed for a specific resource:

```
"OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 02          /* OPSMVS MESSAGE EVENT */
"OI GTMEDIOP" TOKEN 05 IPO1        /* SMF ID */
```

```

"OI GTMEDIOP" TOKEN 03 BATCH001 /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 GTMAUOF /* MESSAGE ID indicating automation off */
"OI GTMEDIOP" TOKEN 49 "Automation ending for object BATCH001" /* MESSAGE TEXT*/
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF

```

The following example notifies Event Pump that the automation procedure ended for a specific resource:

```

"OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 02 /* OPSMVS MESSAGE EVENT */
"OI GTMEDIOP" TOKEN 05 IPO1 /* SMF ID */
"OI GTMEDIOP" TOKEN 03 BATCH001 /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 GTMAUAB /* MESSAGE ID indicating automation off */
"OI GTMEDIOP" TOKEN 49 "Automation terminating for object BATCH001" /*MESSAGE TEXT*/
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF

```

Additional examples of requests

The following examples show additional requests that you can make with the CA OPS/MVS event feed.

CA OPS/MVS

```

/* Pass MESSAGE to Object */
ADDRESS "TSO" "OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 02 /* OPSMVS MESSAGE EVENT */
"OI GTMEDIOP" TOKEN 05 IPO1 /* SMF ID */
"OI GTMEDIOP" TOKEN 03 CICSREG1 /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 SAM001I /* MESSAGE ID */
"OI GTMEDIOP" TOKEN 49 "Test message from OPSMVS" /* MSGTXT */
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF

/* Pass EXCEPTION to Object */
"OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 03 /* OPSMVS EXCEPTION EVENT */
"OI GTMEDIOP" TOKEN 05 IPO1 /* SMF ID */
"OI GTMEDIOP" TOKEN 03 CICSREG2 /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 TMAX /* EXCEPTION */
"OI GTMEDIOP" TOKEN 49 500 /* EXCEPTION VALUE */
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF

/* Pass IMS data */
"OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 18 03 /* IMS MESSAGE EVENT */
"OI GTMEDIOP" TOKEN 02 IMSS /* IMS OBJECT TYPE */
"OI GTMEDIOP" TOKEN 15 USIBMNT.NTA7MVS /* NETWORK.APPLID */
"OI GTMEDIOP" TOKEN 16 IPO1 /* SMF ID */
"OI GTMEDIOP" TOKEN 17 IMA1CTL /* IMS SUBSYSTEM NAME */
"OI GTMEDIOP" TOKEN 63 THRESHOLD /* STATE OF OBJECT */
"OI GTMEDIOP" TOKEN 64 RM0710W /* MESSAGE ID */
"OI GTMEDIOP" TOKEN 65 "Long message queue > x % ..." /*TEXT*/
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF

/* Pass OPC Exception data */
"OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 05 /* OPC Exception EVENT */
"OI GTMEDIOP" TOKEN 03 OPCTST01 /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 05 IPO1 /* SMF ID */
"OI GTMEDIOP" TOKEN 08 'BATCHREPHR' /* EXCEPTION */
"OI GTMEDIOP" TOKEN 10 OPCTST01 /* OPC APPLICATION NAME */
"OI GTMEDIOP" TOKEN 11 000406005 /* OPC INPUT ARRIVAL TIME */
"OI GTMEDIOP" TOKEN 12 CPU1 /* OPC WORKSTATION ID */
"OI GTMEDIOP" TOKEN 49 'REPLY FOR JOB OPCTST01' /* MSG TEXT */
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF

```

Error handling

When the EDI cannot send data to Event Pump, the EDI gets a return code of greater than 0 and a GTMxx101 message is sent to Netcool/OMNIBus. When the EDI failure is resolved, a GTMxx102 message is sent to Netcool/OMNIBus. See “API exit routine return codes” on page 176 for a list of return codes.

Modifying the source

To support the passing of data from a source product or application to Event Pump through the EDI, make the following modifications to the application capture within the originating external product or application.

Pass the data in the following order:

1. START_OF_DATA
2. Format type and action type
3. Data type and data values
4. END_OF_DATA

See “CA OPS/MVS (format type 09)” on page 178 for information about specific format types and action types.

The following example describes how to pass a state change using the EDI:

```
"OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 01          /* OPSMVS STATE CHANGE EVENT */
"OI GTMEDIOP" TOKEN 05 IPO1        /* SMF ID */
"OI GTMEDIOP" TOKEN 06 SUBSYSTEM  /* RESOURCE TYPE */
"OI GTMEDIOP" TOKEN 03 STCTEST1    /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 ABENDING    /* STATE CHANGE */
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF
```

Example notes:

1. The TOKEN unique value is used to distinguish between concurrent calls to the EDI. With the use of the TOKEN value, data is not lost between calls to the EDI.
2. The first parameter that is passed is START_OF_DATA. This parameter initializes the EDI to start accepting information.
3. The second parameter that is passed contains the format type and action type. In this case, the format type is 09 and the action type is 01. You can pass the format type with or without the action type in numeric or character format.

The second parameter can also be one of the following format types:

- "OI GTMEDIOP" 09 STATE_CHANGE
- "OI GTMEDIOP" OPSMVS STATE_CHANGE
- "OI GTMEDIOP" OPSMVS 01

4. The next parameters that are passed contain data type and data:

Table 13. Parameters that contain data type and data

Data type	Data
05 (SMFID)	IPO1 (name of MVS image)
06 (RESOURCE_TYPE)	SUBSYSTEM
03 (OBJECT_NAME)	STCTEST1
08 (EVENT_TYPE)	ABENDING

5. END_OF_DATA and TRACEOFF are the last parameters that are passed. END_OF_DATA notifies EDI that there are no more records to pass. TRACEOFF informs the EDI not to print out the record that is sent to Event Pump. If TRACEON is passed, the

record that is passed to Event Pump is printed. This record is useful for troubleshooting a problem with the EDI.

Usage notes:

1. Format, action, and data type are passed in their numeric or character formats.
2. Use single quotation marks (') when passing text, for example, message text. The following example posts the Test message from OPSMVS message with a message ID of SAM001I to the BATCH001 resource:

```
"OI GTMEDIOP" START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TOKEN 09 02      /* OPSMVS MESSAGE EVENT */
"OI GTMEDIOP" TOKEN 05 IP01    /* SMF ID */
"OI GTMEDIOP" TOKEN 03 BATCH001 /* OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 SAM001I /* MESSAGE ID */
"OI GTMEDIOP" TOKEN 49 "Test message from OPSMVS" /* MESSAGE TEXT */
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF
```

3. An optional TRACEPROG parameter is also available to trace the entire EDI program (GTMEDIOP). You can trace all call statements or specific ones.

An example of tracing all call statements follows:

```
"OI GTMEDIOP" TRACEPROG 'START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TRACEPROG TOKEN 09 02      /*OPSMVS MESSAGE EVENT */
"OI GTMEDIOP" TRACEPROG TOKEN 05 IP01    /*SMF ID */
"OI GTMEDIOP" TRACEPROG TOKEN 03 BATCH001 /*OBJECT NAME */
"OI GTMEDIOP" TRACEPROG TOKEN 08 SAM001I /*MESSAGE ID */
"OI GTMEDIOP" TRACEPROG TOKEN 49 "Test message from OPSMVS " /*MESSAGE TEXT*/
"OI GTMEDIOP" TRACEPROG TOKEN END_OF_DATA TRACEOFF
```

Following is an example of tracing specific call statements:

```
"OI GTMEDIOP" 'START_OF_DATA
Token = "GTM" || rc
"OI GTMEDIOP" TRACEPROG TOKEN 09 02      /*OPSMVS MESSAGE EVENT */
"OI GTMEDIOP" TOKEN 05 IP01              /*SMF ID */
"OI GTMEDIOP" TRACEPROG TOKEN 03 BATCH001 /*OBJECT NAME */
"OI GTMEDIOP" TOKEN 08 SAM001I           /*MESSAGE ID */
"OI GTMEDIOP" TOKEN 49 "Test message from OPSMVS " /*MESSAGE TEXT */
"OI GTMEDIOP" TOKEN END_OF_DATA TRACEOFF
```

Chapter 13. BMC MainView

Event Pump processes events from the following BMC products and sends them to Netcool/OMNIBus where they are stored in the Netcool/OMNIBus Object Server.

- “BMC MainView for z/OS”
- “BMC MainView for CICS” on page 77
- “BMC MainView for DB2” on page 79

BMC MainView for z/OS

To enable Event Pump to capture BMC MainView for z/OS Exception Monitor sampler warning messages, customize BMC MainView for z/OS to use the Write to Operator (WTO) macro to direct these messages to the system console. The system console is monitored by Event Pump for z/OS. Event Pump uses registration to create internal message traps to detect when specific messages are displayed on the console, and Event Pump then takes appropriate action. To enable Event Pump to detect and act on messages from BMC MainView for z/OS Exception Monitor samplers, the monitor member for the customizer creates the appropriate message traps to capture these messages, and these messages are then processed by the customizer. The customizer translates the messages into EIF events and then routes them to the Netcool/OMNIBus Server.

Table 14 provides an overview of the tasks that are required to configure BMC MainView for z/OS to interface with Event Pump.

Table 14. Implementing the BMC MainView for z/OS as an event feed

Task	Where to find information
Direct BMC MainView for z/OS messages to the z/OS system log	“Configuring BMC MainView for z/OS” on page 76
Verify that the BMC MainView for z/OS REXX executable files, messages, and sample members were installed	“Installing Event Pump for BMC MainView z/OS components” on page 76
Register and enable BMC MainView for z/OS for event processing	“Registering BMC MainView for z/OS” on page 76

Event flow

The following process describes how BMC MainView for z/OS exception messages are captured and routed to Netcool/OMNIBus:

1. When a performance indicator that is being monitored by a BMC MainView for z/OS Exception Monitor sampler exceeds its threshold, the sampler uses the WTO macro to issue a warning message to the z/OS system console.
2. Event Pump recognizes the message ID and passes the message to the customizer for processing.
3. The customizer uses various fields from the message to create a unique exception key. An exception record is created and routed to the Netcool/OMNIBus Server.
4. The customizer sets a resolution interval (INTERVAL statement) in the ZEVENT block to ensure that the exception is resolved by the Netcool/OMNIBus.

Event scenario

This event scenario is for a started task that was swapped out for too long a time. The scenario assumes that you performed the BMC MainView for z/OS customization that is described in “Configuring BMC MainView for z/OS.” The scenario also assumes that the BMC MainView for z/OS Exception Monitor Jobs Swapped Out (OUT) sample is active, and that Event Pump is monitoring BMC MainView for z/OS messages.

1. The OUT sampler detects that a started task was swapped out of memory for a time longer than the user-defined threshold, and issues the following WTO message:
PWSOUT01 *WARNING* STC TCAS (31) SWAPPED OUT 778 MINS FOR DETECTED WAIT
2. Event Pump starts the customizer for these key fields in the message:
 - a. A message ID of PSWOUT01
 - b. A resource type of OSTs (from STC, or started task)
 - c. A resource name of TCAS
3. The customizer routes the key fields of this exception to the Netcool/OMNIbus Server.
4. The customizer sets a resolution interval (INTERVAL statement) in the ZEVENT block to ensure that the exception is resolved by Netcool/OMNIbus.

Configuring BMC MainView for z/OS

Configure BMC MainView for z/OS to direct messages to the z/OS system log by inserting the following control statement in the \$\$INSYS0 member of the BBPARM data set:

```
LCLAEW=BOTH
```

For more information about directing messages to the z/OS system log, refer to the BMC MainView for z/OS library.

Installing Event Pump for BMC MainView z/OS components

Make sure that the BMC MainView REXX files, messages, and sample members were installed as part of the SMP/E installation. For installation instructions, see the Program Directory for IBM Tivoli Event Pump for z/OS.

Registering BMC MainView for z/OS

To register BMC MainView for z/OS for monitoring, add the following statement to the REGDATA (GTMSAM22) member:

```
GROUP MainViewZOS      /* The group name for BMC MainView for z/OS */
MONITOR TYPE=MainView ID=MVZOS SUBSYS=ZOS  /* Continuation of group name*/
CONFIG=MZOSPARM /* The default configuration member */
```

Enabling event processing

To enable BMC MainView for z/OS event processing, issue this command:

```
F pump,REGISTER MainViewZOS
```

Registration is performed on the GROUP name, for example MainViewZOS.

Disabling event processing

To disable BMC MainView for z/OS event processing, issue this command:

```
F pump,REGISTER MainViewZOS OFF
```

Registration is performed on the GROUP name.

BMC MainView for CICS

BMC MainView for CICS provides monitors, sometimes called timer-driven requests, that regularly collect data about conditions in a CICS region. Each monitor collects data for a different performance statistic, such as the average response time. When you start a monitor, you define a warning threshold value for the statistic. If the statistic exceeds the warning threshold, the monitor issues a warning message. If the statistic drops beneath the warning threshold, then the service issues a matching resolution message.

Integrating BMC MainView for CICS with Event Pump includes capturing these warning and resolution messages, translating them into events, and passing the events to Netcool/OMNIBus.

BMC MainView for CICS monitors issue messages for two types of events: when a performance statistic exceeds its threshold and when a performance statistic drops beneath its threshold. To enable Event Pump to capture these events, you must customize BMC MainView for CICS monitors to use the Write to Operator (WTO) macro to direct their messages to the z/OS system console. The z/OS system console is monitored by Event Pump.

To enable Event Pump to detect and act on messages from BMC MainView for z/OS Exception Monitor samplers, the monitor member for the customizer creates the appropriate message traps to capture these messages, and these message traps are then processed by the customizer. The customizer translates the messages into EIF events, then routes them to the Netcool/OMNIBus Server.

Table 15 provides an overview of the tasks that are required to configure BMC MainView for z/OS to interface with Event Pump.

Table 15. Implementing BMC MainView for CICS as an event feed

Task	Where to find information
Direct BMC MainView for CICS messages to the z/OS system log	"Configuring BMC MainView for CICS" on page 79
Register and enable BMC MainView for CICS for event processing	"Registering BMC MainView for CICS" on page 79

Event flow

The following process describes the event flow for the WTO messages that were captured by Event Pump for z/OS, issued by BMC MainView for CICS monitors, and routed to Netcool/OMNIBus:

1. A BMC MainView for CICS monitor uses the WTO macro to issue a warning message to the z/OS system console when a performance statistic exceeds its threshold or a resolution message when a performance statistic drops beneath its threshold. These message IDs have the formats shown in Table 16, depending on the particular BMC MainView for CICS monitor (as described in the MainView for CICS library), where *nnn* is a 3-digit number that identifies the monitor that issued the message.

Table 16. Message Formats for BMC MainView for CICS

Message ID	Description
FTnnn0W	FTnnn0W Warning message
FTnnn1I	FTnnn1I Matching resolution message

Table 16. Message Formats for BMC MainView for CICS (continued)

Message ID	Description
FTnnnW	The warning message from @DB2NDB2/IMS not attached where nnn is 517 or @IMSN nnn is: 518 BMC MainView for CICS resource monitors
FTnnnI	Matching resolution message
FTnnnX	The warning message where x indicates the severity: I for informational W for warning S for severe
FTnnnZ	Matching resolution message

2. The Event Pump message traps recognize the message ID and pass the message to the customizer for processing.
3. The customizer translates the BMC MainView for CICS warning or information message into an Event Pump exception or resolution event. Event Pump event IDs are the same as the BMC MainView for CICS warning message IDs, but without the letter W (or, sometimes, S or I) suffix. For example, the FT1020W warning message becomes the FT1020 exception.
4. Event Pump stores the event in the data space.
5. Event Pump retrieves the event and forwards it to Netcool/OMNIBus.

Event scenarios

The following scenarios illustrate the event processing for BMC MainView for CICS when the DB2 CICS interface is unavailable for the C1H1CICS region.

- “DB2 CICS interface unavailable”
- “DB2 CICS interface becomes available again”

DB2 CICS interface unavailable

1. BMC MainView for CICS detects that the DB2 CICS interface is unavailable for the C1H1CICS region, and issues the following WTO message:
FT517W DB2 INTERFACE unavailable FOR DB2 ID (DB6B),TGT=CIH1CICS
2. Event Pump captures the message, and starts the customizer that translates the FT517W message into an FT517 exception event for the CIH1CICS CICS region resource.
3. The customizer routes the exception event to the CIH1CICS CICS region resource in Netcool/OMNIBus.

DB2 CICS interface becomes available again

1. When BMC MainView for CICS detects that the DB2 CICS interface for the C1H1CICS CICS region is available again, it issues the following WTO message:
FT517I DB2 INTERFACE AVAILABLE FOR DB2 ID (DB6B),TGT=CIH1CICS
2. Event Pump captures the message, and starts the customizer that translates the FT517I message into a resolution event for the earlier FT517 exception.
3. The customizer routes the resolution event to the CIH1CICS CICS region resource in Netcool/OMNIBus

Configuring BMC MainView for CICS

Follow these steps to configure BMC MainView for CICS:

1. Configure BMC MainView for CICS to use the z/OS system log.

Configure your BMC MainView for CICS services to direct their messages to the z/OS system console by specifying WMSG=WT0 when you start the service. For details on specifying keywords when starting monitors, see the appropriate document in the BMC MainView for CICS library.

2. Configure BMC MainView for CICS to Specify Correct Target.

The CICS region relating to the BMC MainView message must be in the message text. To ensure that the CICS region relating to the BMC MainView message is in the message text, use the TARGET parameter in the BMC MainView BBIJNTxx and BBIISPxx members and specify the CICS region job or started task name. For details on specifying the TARGET parameter, see the BMC MainView for CICS library.

Registering BMC MainView for CICS

To register BMC MainView for CICS for monitoring, add the following statement to the REGDATA (GTMSAM22) member.

```
GROUP MainViewCICS /* The group name for BMC MainView for CICS */  
MONITOR TYPE=MainView ID=MVCICS SUBSYS=CICS /* Continuation of the group name */  
CONFIG=MCICPARM /* The default configuration member */
```

Enabling event processing

To enable BMC MainView for CICS event processing, issue this command:

```
F pump,REGISTER MainViewCICS
```

Registration is performed on the GROUP name, for example MainViewCICS.

Disabling event processing

To disable BMC MainView for CICS event processing, issue this command:

```
F pump,REGISTER MainViewCICS OFF
```

Registration is performed on the GROUP name.

BMC MainView for DB2

BMC MainView for DB2, a software product from BMC Software, provides services that monitor DB2 performance indicators. Each service monitors a single indicator, such as the percentage of buffer pool usage. When you start a service, you can define a threshold value for its indicator. If the indicator exceeds the threshold value, the service issues a warning message. If the indicator drops beneath the threshold value, the service issues a matching condition-clear message.

Integrating BMC MainView for DB2 with Event Pump includes capturing these warning and condition clear messages and translating them into events that are understood by Netcool/OMNIBus. This integration includes the following types of BMC MainView for DB2 services:

- Resource Monitors
- Workload Monitors
- Background Exception Samplers

BMC MainView for DB2 Resource Monitor, Workload Monitor, and Background Exception Sampler services issue messages for following types of events:

- A performance indicator exceeds its threshold.
- A performance indicator drops beneath its threshold.

To enable Event Pump to capture these events, see “Registering BMC MainView for DB2” on page 82.

Table 17 provides an overview of the tasks that are required to configure BMC MainView for z/OS to interface with Event Pump.

Table 17. Implementing BMC MainView for DB2 as an event feed

Task	Where to find information
Direct BMC MainView for DB2 messages to the z/OS system log	“Configuring BMC MainView for DB2” on page 81
Register and enable BMC MainView for DB2 for event processing	“Registering BMC MainView for DB2” on page 82

Event flow

The following event flow describes how Event Pump captures BMC MainView for DB2 WTO messages, and routes them to Netcool/OMNIBus:

1. A BMC MainView for DB2 service uses the WTO macro to issue a warning message to the z/OS system console when a performance indicator exceeds its threshold. A BMC MainView for DB2 service also uses the WTO macro to issue a condition clear message when a performance indicator drops beneath its threshold as shown in Table 18, where *nnn* is a 3-digit number that identifies the service that issued the message.

Table 18. BMC MainView for DB2 Message IDs

Message ID	Description
DSnnn0W	Warning message from a Resource Monitor service
DSnnn1I	Warning message from a Resource Monitor service
DWnnn0W	Warning message from a Workload Monitor service
DWnnn1I	Condition clear message from a Workload Monitor service
DZnnn0A	Warning message from a Background Exception Sampler service where the a identifies the severity of the warning: I for informational W for warning S for severe
DZnnn1I	Condition clear message from a Background Exception Sampler service

2. Event Pump trap processing recognizes the message ID, and passes the message to a customizer handler.
3. The customizer translates the BMC MainView for DB2 warning or condition clear message into an Event Pump exception or resolution event. Event Pump exception names are like the BMC MainView for DB2 warning message ID (in some cases, without the I, W, or S severity suffix; or appended by an underscore and a parameter, to help further identify the source of the event). For example, the DZ0530W warning message becomes the DZ0530 exception in Event Pump, and the DZ0531I message, which is the matching information message, becomes a resolution event that removes the exception.
4. The Event Pump source collector stores the event in the Event Pump data space.

5. The Event Pump event distributor retrieves the event from the Event Pump data space and forwards it to Netcool/OMNIbus Server.

Event scenarios

These are the event scenarios for BMC MainView for DB2:

- “TSO user exceeds its processor resource usage threshold”
- “TSO user processor resource use drops beneath threshold”

TSO user exceeds its processor resource usage threshold

1. A background exception sampler detects that a TSO user ID (TONYF) exceeded its processor resource usage threshold value, and issues the DZ0630W warning message to the z/OS system console:

```
DZ0630W DB6S RUNAWAY TSO QUERY USER TONYF CPU%=16.8 PGM=DSNECP10 PLAN=DSNESPCS BBDB
```
2. Event Pump message processing captures the message, and starts the customizer that translates the DZ0630W message into a DZ0630_TONYF exception event for the DB6S DB2 subsystem object.
3. The customizer routes the exception event to Netcool/OMNIbus for the DB6S DB2 subsystem resource.

TSO user processor resource use drops beneath threshold

1. A background exception sampler detects that the processor resource use for the TONYF TSO user ID dropped beneath the threshold value, and issues the following DZ0631I condition clear message to the z/OS system console: DZ0631I DB6S EXCEPTION CLEARED: RUNAWAY TSO QUERY USER TONYF CPU%=0.0

```
PGM=DSNECP10 PLAN=DSNESPCS BBDB
```
2. Event Pump message processing captures the message, and starts the customizer that translates the DZ0631I message into a resolution event for the earlier DZ0630_TONYF exception.
3. The customizer routes the resolution event to Netcool/OMNIbus for the DB6S DB2 subsystem resource.

Configuring BMC MainView for DB2

Complete the following steps to configure BMC MainView for DB2:

1. Configure BMC MainView for DB2 to use the z/OS SYSLOG event feed:
 - Configure the BMC MainView for DB2 services to direct messages to the z/OS system console by specifying WMSG=WTO when you start the service. For example, when you start the THDQD service, specify:

```
REQ=THDQD WMSG=WTO
```

For details on specifying keywords when starting monitors, see the BMC MainView for DB2 library.

2. Configure BMC MainView for DB2 Background Exception Sampler to use WTO:
 - If you want all exception samplers to use WTO, configure BMC MainView for DB2 Background Exception Sampler services to direct messages to the z/OS system. For example, in the BMC MainView for DB2 DMRBEXxx member specify:

```
BMSGWTO=YES
```
 - To configure your BMC MainView for DB2 Background Exception Sampler services to direct messages to the z/OS system console, specify the following

statement in the BMC MainView for DB2 DMRBEXxx member if you want individual exception samplers to use WTO:

WTO=YES

For more information, see the appropriate document in the BMC MainView for DB2 library.

3. Configure BMC MainView for DB2 to specify the correct target.

The DB2 region relating to the BMC MainView message must be in the message text. To ensure that the DB2 region relating to the BMC MainView message is in the message text, use the TARGET parameter in the BMC MainView BBIJNTxx and BBIISPxx members and specify the DB2 subsystem name.

For details on specifying the TARGET parameter, see the appropriate document in the BMC MainView for DB2 library.

Registering BMC MainView for DB2

To register the BMC MainView for DB2 for monitoring, add the following statement to the GTMINPUT member:

```
GROUP MainViewDB2 /* The group name for BMC MainView for DB2 */  
MONITOR TYPE=MainView ID=MVDB2 SUBSYS=DB2 /* Continuation of the group name */  
CONFIG=MDB2PARM /* The default configuration member */
```

Enabling event processing

To enable BMC MainView for DB2 event processing, issue this command:

```
F pump,REGISTER MainViewDB2
```

Registration is performed on the GROUP name, for example MainViewDB2.

Disabling event processing

To disable BMC MainView for DB2 event processing, issue this command:

```
F pump,REGISTER MainViewDB2 OFF
```

Registration is performed on the GROUP name.

Chapter 14. CICS

Use base CICS Discovery if you have a vendor-acquired CICS monitor, and want to discover CICS resources. Base CICS Discovery is not a requirement for users of the CICSplex System Manager event feed.

Event Pump can monitor base resources and low-level resources. The following base CICS subsystem data resources are monitored:

- CICS-DB2 connections
- CICS Corba servers
- CICS DJARS
- Files
- Logical units (LUs)
- Regions
- transactions

CICS connections are not monitored.

Table 19 provides an overview of the tasks that are required to configure the CICS Transient Data Queues event feed to interface with Event Pump.

Table 19. Implementing CICS Transient Data Queues event feed

Task	Where to find information
Enable Event Pump to collect CICS transient data queue messages.	"Installing the CICS Transient Data Queue Exit program"
Review and update the CICS started task.	"Review the CICS started task" on page 84
Review and update as necessary the EDI_BUFFER_SIZE startup parameter.	"Reviewing the Event Pump GPZPARMS member" on page 84
Register base CICS resources and CICS Transient Data.	"Registering base CICS resources and CICS Transient Data" on page 84

Installing the CICS Transient Data Queue Exit program

The CICS Transient Data Queue Exit program enables the Event Pump to collect CICS transient data queue messages. The External Data Interface (EDI) within Event Pump accepts message data from the CICS program that is using this exit.

To install the Transient Data Queue Exit program, follow these steps:

1. Assemble and install the AOPCICSx program into the CICS program. The AOPCICS3 sample job (CICS Version 3.1 and later) is found in the SGTMSAMP library. By default, queues starting with the letter C are processed. If you want to collect another range of CICS Transient Data Queues, customize the GPZTDQTB sample macro before running the AOPCICS3 sample job.
2. Add the AOPCICSx program to the startup Program List Table (PLT) after the DFHDELIM entry. The PLT is a list of programs that the CICS program runs during startup and shutdown.
3. Optional: Define a transaction code for the AOPCICSx program (4-character name).

By defining a transaction code, when you log on to the CICS program, you can use the transaction code to stop or start the exit.

After the program is installed, the CICS program issues this message at startup:

Review the CICS started task

Review the CICS started task. If you are using the ACC1IDxx statement as described in “Updating the GPZPUMP startup procedure” on page 32, add the following statement:

```
//ACC1IDxx DD DUMMY
```

Reviewing the Event Pump GPZPARMS member

If you are losing data because of the number of CICS regions that are sending data to the data collector, increase the EDI_BUFFER_SIZE parameter. You can also use the Event Pump EDI modify command to increase the buffer size. For more information about the EDI modify command, see *Event Pump for z/OS Command Reference*.

Usage note: In a 3-address-space environment, specify the startup parameters in the PARMPUMP member. The source collector startup procedure is GTMPUMP.

Registering base CICS resources and CICS Transient Data

To register base CICS resources for monitoring, add the following statement to the REGDATA (GTMSAM22) member:

```
MONITOR TYPE=CICS JOBNAME=jobname
```

To register CICS Transient Data for monitoring, add the following additional statements to the REGDATA member:

```
MONITOR TYPE=TDQ
```

For more information about advance monitoring for the CICSplex System Manager environment, see “Registering CICSplex System Management source” on page 88.

Chapter 15. CICSplex System Manager

Event Pump uses the Real Time Analysis (RTA) feature of CICSplex System Manager (SM).

Table 20 provides an overview of the tasks that are required to configure CICSplex SM to interface with Event Pump.

Table 20. Implementing the CICSplex System Manager as an event feed

Task	Where to find information
Review and update as necessary the resource definitions.	"Reviewing resource definitions"
Set up system availability monitoring.	"Defining System Availability Monitoring Alerts" on page 86
Install sample MAS resource monitoring definitions.	"Defining MAS Resource Monitoring Alerts." on page 86
Filter resource and events.	"Excluding resources and events" on page 87
Review and update as necessary the Event Pump GTMPUMP startup procedure.	"Reviewing the GPZPUMP procedure" on page 88
Register CICSplex System Manager as an event feed.	"Registering CICSplex System Management source" on page 88

If you want information about...	Refer to...
Overview of CICSplex System Manager functions	<i>CICSplex SM Concepts and Planning</i>
Use of the CICSplex SM screens	<i>CICSplex SM Administration</i>
Configuration information	<i>CICSplex SM Managing Resource Usage</i>
Information about the use of the web interface	<i>CICSplex SM Web User Interface Guide</i>

Reviewing resource definitions

Event Pump requires the following definitions for CICSplex SM resources:

CICSSYS

Defines the CICS system and associates it with a CICSplex. The CICS regions must be connected to a CMAS region.

RTASPEC

Analysis specification that identifies the default control attributes that are used for system availability monitoring (SAM) and provides an anchor for all analysis definitions and status definitions associated with a CICS system.

RTAGROUP

Specification that points to the RTADEF specifications that are used for MAS Resource Monitoring (MRM).

RTADEF

Analysis definition that defines the evaluations to be performed on a

periodic basis and the actions to be taken if a notifiable condition occurs. For the Event Pump, this definition is used to define transaction and file status reporting. An RTADEF definition has a related EVALDEF definition that is used to define the evaluation and a related ACTNDEF definition to define the action.

EVALDEF

Evaluation definition that identifies the resources that are to be sampled and evaluated. When the result of the evaluation is true, the associated analysis definition (RTADEF) is used to determine if a notifiable condition occurred.

ACTNDEF

Action definition that designates the type of external notification that occurs when the condition or conditions identified in the analysis definition are true. External messages are set to YES.

For information, refer to the *CICSplex SM Concepts and Planning Guide*.

Defining System Availability Monitoring Alerts

The CICSplex SM system availability monitoring (SAM) function monitors CICS systems during their planned hours of availability. If any of a set of predefined conditions occurs while a CICS system is monitored, CICSplex SM sends out an external notification. These conditions are:

SAMOPS	System unavailable
SAMSOS	Short on storage
SAMSDM	System memory dump in progress
SAMTDM	Transaction memory dump in progress
SAMMAX	System at maximum number of tasks
SAMSTL	System in stall condition (deadlock)

To set up for CICSplex SM system availability monitoring:

1. Use the CICSplex SM interactive panels and enter RTASPEC. The RTASPEC specification provides a list of RTASPEC resources defined for your CICSplex.
2. Find the RTASPEC resource corresponding to the MAS (CICS system) for which you are doing the setup. Enter UPD as the action for this RTASPEC.
3. Determine which of the conditions for which you want to generate alerts. For those conditions, specify an action definition (ACTNDEF) in the ACTION field. For CICSplex SM purposes, the severity is not used. The ACTNDEF definition can be an existing definition. If the ACTNDEF definition is used for other purposes, make sure that usage is compatible with usage for Event Pump.
4. For the preceding ACTNDEF definitions, make sure that ALERT is set to YES. To set ALERT to YES, specify ACTNDEF on the CICSplex SM WUI and then create or update for each ACTNDEF definition. For more information about how to set up a SAM function, see the *CICSplex SM Managing Resource Usage Guide*.

Defining MAS Resource Monitoring Alerts.

Use the CICSplex SM MAS resource monitoring (MRM) function to monitor the status of any specific or generic CICS resources. Event Pump specifically monitors the following resources:

- Files
- Transactions

- CICS connections
- DB2 connections

Events that are generated by monitoring other resources are applied to the CICS region that owns them.

Sample MRM definitions for the resources that are monitored by the Event Pump are provided in the SGTMSAMP library. Install these definitions or a set that is functionally equivalent, to monitor the listed resources. The following table lists the sample alert MRM definitions.

Table 21. Sample alert MRM definitions

Sample Name	Description
GPZCDB2C	DB2 connections
GPZCFILE	Files
GPZCGRP	CICS groups
GPZCMAS	CICS regions and connections
GPZCMISC	Miscellaneous resources
GPZCTRAN	Transactions

Edit the sample MRM files and modify the CONTEXT field as required by your installation. You can also modify other fields in these files as required. For each sample, complete the following steps from the CICSplex SM Web User Interface (WUI):

1. From the Batched Repository Update request panel of the Administration views, check the Stopped processing state and click **Execute**.
2. On the Execute panel, enter the following information:
 - Input data set name with the data set name of the sample member
 - Input member name with the name of the sample
 - Print node and Destination user ID with the system node and user ID where output is sent

Click **Execute**. This starts a batch job that is named the same as the CMAS name.

3. To customize APM events for either Groups or Definitions, go to the Administration views, and then navigate to the RTA analysis point monitoring panel. Choose either Groups or Definitions from that panel, select the objects on which you want to operate, and then click **Install analysis specification**. From the Install analysis specification panel, set the Scope value to the target CICS region.

APM events are typically scoped to the CICSplex. This implementation of the CICSplex SM event feed requires that APM events are scoped to a CICS Region. See *CICSplex SM Managing Resource Usage* for further details.

Excluding resources and events

You can use the exclude facility to exclude CICSplex SM from monitoring specified resources and events. The filter uses the CICS name, and for additional filtering, it can also use the subresource type and name. The filter can include specific event identifiers. For additional information, see “Excluding resources and events” on page 94.

The hierarchy for CICSplex SM resources and subresources, as used by Event Pump, is described in Table 22. This hierarchy is used during Event Pump exclude processing. During discovery or event processing, subordinate resources behave differently, depending on the exclude definitions.

Table 22. CICSplex SM Resources and Sub-resources

Type	Description	Resource name #F	Resource type #F	Subresource name #F	Subresource type #F
CICM	CICS CMAS	21	22	N/A	N/A
CIMA	CICS MAS	21	22	N/A	N/A
CITR	CICS remote transaction	21	22	23	24
CITL	CICS local transaction	21	22	23	24
CIFR	CICS remote file	21	22	23	24
CIFL	CICS local file	21	22	23	24

See Table 23 for examples.

Table 23. Examples

TYPE	JOBNAME	RESOURCE	EVENTID	SYSNAME	COMMENT
CIFL	WUINCM	*	*	*	Exclude all events that contain a CICS system field starting with WUINCM
CIFL	WUINCM	EYUWR*	*	*	Exclude all events that contain a local file that starts with EYUWR for a CICS system field starting with WUINCM

Reviewing the GPZPUMP procedure

Make sure that the GTMMON DD statement references the data set where the ICICPARM member is located. By default, the ICICPARM member is in the SGTMSAMP data set.

Note: In a 3-address-space environment, review and update the GTMPUMP startup.

Registering CICSplex System Management source

To register CICSplex System Management for monitoring, add the following statement to the REGDATA (GTMSAM22) member:

```
MONITOR TYPE=CPSM
```

Optionally, you can turn on debugging with the DEBUG=YES parameter:

```
MONITOR TYPE=CPSM  
DEBUG=YES
```

*

An asterisk (*) is a continuation indicator for the registration entries and is coded in position 72.

Chapter 16. DB2 Universal Database for z/OS

Event Pump uses DB2 Universal Database for z/OS messages to monitor the state of a DB2 subsystem and its components.

Messages that are produced by DB2 programs are trapped using Event Pump. The event distributor communication task forwards events to the EIF probe for processing. Events that represent a state change of a monitored resource are sent to Netcool/OMNIBus.

Table 24 provides an overview of the tasks that are required to configure DB2 Universal Database for z/OS to interface with Event Pump.

Table 24. Implementing DB2 Universal Database for z/OS as an event feed

Task	Where to find information
Make sure that the DB2 programs, REXX executable files, messages, and sample members are installed. Review sysplex support and security.	"Installing Event Pump for DB2 components"
Configure each DB2 subsystem or data sharing group.	"Configuring each DB2 subsystem or data sharing group" on page 92
Review and update as necessary the Event Pump GTMPUMP startup procedure.	"Reviewing the GPZPUMP procedure" on page 93
Register and enable DB2 Universal Database for z/OS for event processing.	"Registering the DB2 event feed for monitoring" on page 93
Filter resources and events as necessary for your installation.	"Excluding resources and events" on page 94

Installing Event Pump for DB2 components

Verify that the DB2 programs, REXX executable files, messages, and sample members were installed as part of the SMP/E installation. For installation instructions, see *Program Directory for IBM Tivoli Event Pump for z/OS*.

Adding DB2 sysplex support

In a sysplex configuration, it is possible for the same DB2 subsystem to be started at different times on different images. It is also possible to have separate DB2 subsystems of the same name defined on different images within the sysplex. It is important for Event Pump to distinguish between these two situations.

A DB2 subsystem that can be started on more than one image within a sysplex requires that you select **Participates in Sysplex** from the **Source/390** tab of the Properties page for a DB2 subsystem resource (this value is the default value) on Netcool/OMNIBus. Conversely, if the DB2 subsystem is defined to only one image, then ensure that **Participates in Sysplex** is cleared.

If you selected **Participates in Sysplex** then only a single copy of data objects is kept in the Event Pump database. When the DB2 subsystem is activated in another image, none of the resource types that reside beneath the data sharing resource group are detected (except for new objects that were defined since the last

detection). Instead, only the high-level resource types, for example, the master address space up to the EDM Pool, are detected under the new image. This prevents the creation of redundant data objects during detection. If you select **Participates in Sysplex**, you can improve performance. If you did not select **Participates in Sysplex** in this situation, the subsystem is considered as an entirely new object, and the inclusion of it in any business systems to which it previously belonged is not recognized.

Review security

The DSNR resource class controls access to DB2 subsystems. If this class is enabled, give READ access to the *db2id*.BATCH profiles that represent DB2 subsystems. Event Pump requires this access to monitor the protected DB2 subsystems.

Make sure that the Event Pump user ID has the authority to issue DB2 DISPLAY commands, including the DISPLAY DB and DISPLAY GROUP commands. If DB2 SQL discovery is required, make sure that the Event Pump user ID has the authority to read SYSIBM tables.

Configuring each DB2 subsystem or data sharing group

The following configuration is required for each DB2 subsystem or data sharing group:

- Make sure that each z/OS host in the sysplex where a DB2 data sharing group member can run is also running an Event Pump DB2 event feed that is enabled.
- Make sure that the Global Resource Serialization configuration does not modify the parameters of the enqueue.
- Event Pump uses the DSNREXX DB2 REXX interface to issue SQL commands from a REXX procedure. The DSNREXX interface must be bound to each DB2 subsystem where a discovery is required. For information about how to bind the interface for DB2 REXX Language Support using the DSNTIJRX job, see the DB2 library.
- Run the GPZBTBND job for each combination of a standalone DB2 subsystem (or single member of a DB2 data sharing group) and Event Pump to monitor that DB2 subsystem or DB2 data sharing group. See the comments in the job for information about how to customize the JCL to meet your site standards and for possible changes to the DB2 connection DSN3SATH exit sample. Consult your system administrator and other systems personnel before running this job.

If the same DB2 subsystem or data sharing group is monitored from different Event Pump programs and these Event Pump programs have different Event Pump started task user IDs, run the GRANT step in the GPZBTBND job separately, using each of the Event Pump started task user IDs as the authorization ID.

Examples:

- **Example 1:** Standalone Subsystems (Event Pumps with identical started task user IDs)

Run the GPZBTBND job once for the DB2X subsystem, using an authorization ID of EPUSER1 if the following conditions exist:

- Event Pump on SYS1 is configured to support the DB2X subsystem for DB2 (Event Pump started task user ID is EPUSER1).
- Event Pump on SYS2 is configured to also support the DB2X subsystem for DB2 (Event Pump started task user ID is EPUSER2).

- **Example 2:** Standalone subsystems (Event Pumps with different started task user IDs)

If the following conditions exist, run the GRANT step in the GPZBTBND job twice for the DB5B subsystem, once using an authorization ID of EPUSER11 and once using an authorization ID EPUSER22:

- Event Pump on SYS11 is configured to support the DB5B subsystem for DB2 (Event Pump started task user ID is EPUSER11)
- Event Pump on SYS2 is also configured to support the DB5B subsystem for DB2 (Event Pump started task user ID is EPUSER22)
- **Example 3:** Data Sharing Subsystems (Event Pumps with identical started task user IDs)

If the following conditions exist, run the GPZBTBND job once using authorization ID of EPUSER3:

- Event Pump on SYS3 is configured to support the DB6A subsystem for DB2 (Event Pump started task user ID is EPUSER3)
- Event Pump on SYS4 is configured to support the DB6B subsystem for DB2 (Event Pump started task user ID is EPUSER3)

You can run the GPZBTBND job for either DB6A or DB6B.

- **Example 4:** Data Sharing Subsystems (Event Pumps with different started task user IDs)

If the following conditions exist, run the GRANT step in the GPZBTBND job twice, once using an authorization ID of EPUSER33 and another using an authorization ID of EPUSER44:

- Event Pump on SYS33 is configured to support the DB7A subsystem for DB2 (Event Pump started task user ID is EPUSER33)
- Event Pump on SYS44 is configured to support the DB7B subsystem for DB2 (Event Pump started task user ID is EPUSER44)

You can run the GPZBTBND job for either DB7A or DB7B.

Reviewing the GPZPUMP procedure

Make sure that the GTMMON DD statement references the data set where the IDB2PARM member is located. By default, the IDB2PARM member is in the SGTMSAMP data set.

Note: In a 3-address-space environment, review and update the GTMPUMP startup.

Registering the DB2 event feed for monitoring

To register the DB2 event feed for monitoring, add the following statement to the REGDATA (GTMSAM22) member:

```
MONITOR TYPE=DB2
```

Enabling DB2 event processing

To enable DB2 event processing, issue this command:

```
F pump,REGISTER
```

Disabling DB2 event processing

To disable DB2 event processing, either change the exclude filter to exclude all DB2 subsystems, or issue this command:

```
F pump,REGISTER DB2 OFF
```

When a DB2 subsystem shuts down, and no other DB2 subsystems are active for the same DB2 Datasharing Group, all the related resources are unavailable.

Excluding resources and events

Use the GPZ\$EXCL member in the GTMMON data set to include or exclude subsystems and low-level resources. By default, all subsystems and low-level resources are included. For examples on excluding resources from monitoring, see the comments and examples in the GPZ\$EXCL member.

Supported resource monitoring

Table 25 shows the supported subsystem data resources for the DB2 event feed for the resource monitoring process.

Table 25. Subsystem data resources (DB2)

Base resources	IBM DB2
Subsystem	Yes
Master address space	Yes
IRLM	Yes
Database services address space	Yes
Distributed Data Facility address space	Yes
Stored Procedures address space	Yes
Logs	Yes
Buffer pools	Yes
EDM pool	Yes
Low-level resources	
Data sharing group	Yes
Databases	Yes
Tablespaces	Yes
Tables	Yes
Indexes	Yes
Tablespace partitions	Yes
Index partitions	Yes
View	Yes

Reasons for excluding resources or events

These are some situations when you might not want to monitor some DB2 resources or events:

- Monitoring of test and development systems is not required. You can exclude these systems from Event Pump monitoring.
- Only monitor critical applications (but not others) within a DB2 subsystem.
- You can choose not to monitor exception events from the automatic stopping of databases by nightly batch processing. You can exclude these databases at the start of batch processing, and include them again at batch end.
- Only include subsystems to be monitored, not databases, table spaces, indexes, and tables. You can exclude resources not required for monitoring.

How to exclude resources or events

To exclude DB2 subsystems or resources, edit the GPZ\$EXCL member of the SGTMSAMP library, using the format that is explained in the member. This member is automatically loaded whenever the Event Pump product is started or the GPZ#EXCL command is run. Run the Event Pump GPZ#EXCL command to view the exclude list or to refresh it from the GPZ\$EXCL member or from another GTMMON member.

Hierarchy of exclude processing

The hierarchy for DB2 subsystems and resources, as used by Event Pump, is described in Table 26. This hierarchy is important for the Event Pump exclude processing. During discovery or event processing, subordinate resources behave differently, depending on the exclude definitions:

Table 26. Hierarchy of exclude processing

Exclude activated during Rediscovery		
Resources defined to be excluded	Resources that are excluded	Resources that are not excluded
Subsystem	All resources	
Data Sharing Group – for standalone subsystem	Not applicable	All resources
Data Sharing Group – for data sharing subsystems	DG and all subordinate resources	High-level resources. See Note 3 on page 96.
Database	DB,TS,TB,IX,TP,IP Dependent tables of views	High-level resources VW Dependent views of views
Tablespace	TS,TB,IX,TP,IP Dependent tables of views	High-level resources DB,VW Dependent views of views
Table	TB,IX,TP,IP Dependent tables of views Dependent views of views	High-level resources DB,TS,VW
Index	IX	High-level resources DB,TS,TB,TP,IP,VW Dependent tables of views Dependent views of views
Tablespace Partition	TP,IP	High-level resources DB,TS,TB,IX,VW Dependent tables of views Dependent views of views
Index Partition	IP	High-level resources DB,TS,TB,IX,TP,VW Dependent tables of views Dependent views of views
View	VW Dependent tables of views Dependent views of views	High-level resources DB,TS,TB,IX,TP,IP

Exclude activated during Event Processing		
Subsystem	All resources	See Note 2 on page 96.
Data Sharing Group – for standalone subsystem	Not applicable	All resources

Data Sharing Group – for data sharing subsystems	DG and all subordinate resources	High-level resources. See Note 3.
Database	DB,TS,TP	High-level resources IX,IP
Tablespace	TS	DB,IX,TP,IP
Table	Not applicable. See Note 4.	
Index	IX	High-level resources DB,TS,TP,IP
Tablespace Partition	TP	High-level resources DB,TS,IX,IP
Index Partition	IP	High-level resources DB,TS,IX,TP
View	Not applicable. See Note 4.	
Notes: 1. High-level resources: Subsystem, IRLM, Buffer Pool, EDM Pool, and Logs. 2. If you exclude a DB2 subsystem that participates in a data sharing group, the lower-level resources are still monitored (unless all DB2 subsystems in that data sharing group have been excluded). 3. If exclude filtering is applied at the data sharing group level, the same exclude definition is activated across all NetView domains where DB2 subsystems that participate in the sysplex are to be activated. 4. No events are ever sent for Tables and Views. 5. Excluding a DB2 subsystem or data sharing group reduces resource utilization on both the host and Windows servers. Excluding lower-level objects or events only reduces resource utilization on the Windows servers.		

Refreshing exclude rules

Refresh the exclude rules by using the following command:

```
f pump, REFRESH member
```

In this example, *member* is an optional member name holding the exclude rules.

Excluding at the subsystem level requires additional administrator action. If the subsystem is to be excluded, running this command does not automatically cause the subsystem to be excluded. Only a DB2 subsystem startup can cause a subsystem exclude to work for the rediscovery and subsequent event processing. Even manually starting a rediscovery by running the IHS\$RDIS command does not cause the subsystem to be excluded.

Show a list of Exclude Rules

To display the exclude rules that are currently being used, issue the following SHOW command:

```
GPZ#EXCL SHOW
```

Syntax of the Exclude Rules

```
0  0    1    1    2    2    3    3    4    4    5
1---5---0---5---0---5---0---5---0---5---0
* This is a comment
word1    word2    word3    word4    word5
```

A non-blank in column 1 indicates a comment line

Syntax:

word1 = Object Type

word2 = resource name

word3 = sub-resource name (default = *)

word4 = EVENT ID (default = *)

word5 = SYSNAME (default = *)

The preceding fields can include wildcards for additional matching criteria.

* = any character wildcard

? = single character wildcard

The SYSNAME parameter enables multiple systems reading the same GPZ\$EXCL member and apply common filters (for example, when asterisk (*) is specified) or filters specific to a host.

The exclude filter applies to all events that contain fields that match the filter criteria.

- The DB2 feed generates format code 15 events.
- The F# column in the following table denotes the field number that is used for each object type as defined in the simplified event format.

Specifying complete Object Types (Word 1) instead of more generic types (for example, DBSS instead of an asterisk) might improve performance.

Event fields that are used for exclude matching:

Table 27. Event fields used for exclude matching

Object type	Relative position
Word1 – subsystem level low-level sub-resources	Field 22 Field 24
Word2 – resource name (jobname)	Field 21
Word3 – sub-resource name	Field 23
Word4 – Event Id	Field 18
Word5 – System Name	Field 04

Table 28. DB2 Resources and Sub-resources

Type	Description	Resource Name F#	Resource Type F#	Subresource Name F#	Subresource Type F#
DBDG	DB2Data Sharing Group	21	22	n/a	n/a
DBSS	DB2 Subsystem	21	22	n/a	n/a
DBMS	DB2 Master Address	21	22	n/a	n/a
All low-level resources					
DBLK	DB2 Lock Manager	21	22	23	24

Table 28. DB2 Resources and Sub-resources (continued)

Type	Description	Resource Name F#	Resource Type F#	Subresource Name F#	Subresource Type F#
DBDS	DB2 Database Services	21	22	23	24
DBDD	DB2 DDF	21	22	23	24
DBSP	DB2 Stored Procedures Mgr	21	22	23	24
DBBP	DB2 Buffer Pool	21	22	23	24
DBEP	DB2 EDM Pool	21	22	23	24
DBLG	DB2 Logs	21	22	23	24
DBDB	DB2 Database	21	22	23	24
DBTS	DB2 Tablespace	21	22	23	24
DBTS	DB2 Table	21	22	23	24
DBIX	DB2 Index	21	22	23	24
DBTP	DB2 Tablespace Partition	21	22	23	24
DBIP	DB2 Index Partition	21	22	23	24
DBVW	DB2 View	21	22	23	24
DBSS	DB8*	*	*	*	24

Examples:

Table 29. Examples

Type	Description	Resource Name F#	Resource Type F#	Subresource Name F#	Subresource Type F#
DBSS	DB8*	*	*	*	Exclude all events that contain a DB2 subsystem.
DB*	??X*	*	*	*	Exclude all events that contain a DB2 subresource fields that have a X as its third character.

Table 29. Examples (continued)

Type	Description	Resource Name F#	Resource Type F#	Subresource Name F#	Subresource Type F#
DBTS	D91*	USER*	*	*	Exclude all events that contain a tablespace resource that starts with USER for subsystems starting with D91.

DB2 Resource Hierarchy

Table 30. DB2 Resource Hierarchy

Resource Type	Resource Hierarchy
DBSS	OS+DBSS
DBMS	OS+DBSS+DBMS
DBDG	OS+DBSS+DBDG
DBLK	OS+DBSS+DBLK
DBDS	OS+DBSS+DBDS
DBDD	OS+DBSS+DBDD
DBSP	OS+DBSS+DBSP
DBBP	OS+DBSS+DBBP
DBEP	OS+DBSS+DBEP
DBLG	OS+DBSS+DBLG
DBDB	OS+DBSS+DBDG+DBDB
DBTS	OS+DBSS+DBDG+DBDB+DBTS
DBTB	OS+DBSS+DBDG+DBDB+DBTS+DBTB
DBIX	OS+DBSS+DBDG+DBDB+DBTS+DBTB+DBIX
DBTP	OS+DBSS+DBDG+DBDB+DBTS+DBTB+DBTP
DBIP	OS+DBSS+DBDG+DBDB+DBTS+DBTB+DBTP+DBIP
DBVW	OS+DBSS+DBDG+DBDB+DBVW

Chapter 17. Health Checker for z/OS

You can configure z/OS components, individual products, or independent software vendor (ISV) software to conduct important state and status checks that are compatible with the IBM Health Checker for z/OS. You can use Event Pump to monitor IBM Health Checker check results.

Table 31 provides an overview of the tasks that are required to configure integration with Health Checker for z/OS monitoring.

Table 31. Implementing Health Checker for z/OS

Task	Where to find information
Register the IBM Health Checker application for monitoring.	"Registering Health Checker data feed for monitoring"
Set up filtering and assign severities.	"Assigning Health Checker filtering and severities"

Registering Health Checker data feed for monitoring

To register the IBM Health Checker event feed for monitoring, add the following statement to the REGDATA (GTMSAM22) member:

```
SOURCE TYPE=HCHECK CONFIG=config_member
```

Assigning Health Checker filtering and severities

IBM Health Checker conducts cycles of defined checks at intervals that you specify. Assign one of the following severity levels to the checks, based on the type of problem that is identified in the check and on its level of criticality:

SEVERITY(HIGH)

Indicates that the check routine is checking for high-severity problems in an installation. All exception messages that the check issues with the HZSFMSG service are issued to the console as critical action messages.

SEVERITY(MEDIUM)

Indicates that the check routine is looking for problems that degrade the performance of the system. All exception messages that the check issues with the HZSFMSG service are issued to the console as action messages.

SEVERITY(LOW)

Indicates that the check routine is looking for problems that do not affect the system immediately, but that must be investigated. All exception messages that the check issues with the HZSFMSG service are issued to the console as informational messages.

The severity levels that you assign are mapped to the following Event Pump filters:

Table 32. Event Pump filters

IBM Health Checker severity level	Event Pump filter
HIGH	CRITICAL
MEDIUM	EVENTUAL

Table 32. Event Pump filters (continued)

IBM Health Checker severity level	Event Pump filter
LOW	INFORMATIONAL

The *check status* describes the output of the check when it last ran. These check statuses are mapped to the following Netcool/OMNIBus severities:

Table 33. Netcool/OMNIBus severities

IBM Health Checker check status	Netcool/OMNIBus severity
The check found a high severity exception.	EXCEPTION-HIGH
The check found a medium severity exception.	EXCEPTION-MEDIUM
The check found a low severity exception.	EXCEPTION-LOW
The check encountered a parameter error.	PARAMETER ERROR
The check was successful.	SUCCESSFUL

You can customize IBM Health Checker filtering and severities in the *config_member* data set member that you specified in the REGDATA member. The configuration member is specified in the concatenation of the GTMMON DD statement in the GPZPUMP procedure.

Usage note: In a 3-address-space environment, the source collector startup procedure is GTMPUMP.

The configuration member has the following format:

```
[FILTER=filter] --- Optional The level of filtering to be preformed
[EXECPHIGH=excephigh] --- Optional Severity assigned to Exception-High states
[EXECPMED=excpmed] --- Optional Severity assigned to Exception-Medium states
[EXCEPLOW=exceplow] --- Optional Severity assigned to Exception-Low states
[PARMERROR=parmerror] --- Optional Severity assigned to parameter errors
```

For an explanation of the configuration statements, see *Event Pump for z/OS Configuration Reference*.

Here is an example of the configuration member for Health Checker (IHCKPARM sample):

```
*
* Monitor Health Check for z/OS
*
FILTER=EVENTUAL
EXCEPHIGH=CRITICAL
EXCEPMED=MINOR
EXCEPLOW=WARNING
PARMERROR=MAJOR
```

In this example, the check result with a WTOTYPE value of EVENTUAL and CRITICAL is processed. A check result with a WTOTYPE value of INFO is ignored. The assigned status for the exception types is based on the check status from the last run.

Chapter 18. IMS

Event Pump uses IMS messages and exception thresholds to monitor the state of IMS subsystems and related systems (IRLM, CQS, FDBR, IMS Connect, OM, RM, SCI, dependent regions, and any IMS resources such as databases, programs, transactions, or logs) using the Tivoli NetView for z/OS product.

Event Pump can monitor base resources and low-level resources.

The following base IMS subsystem data resources are monitored:

- BMP / JBP regions
- CQS region
- FDBR region
- IFP regions
- IMS Control region
- IMS - DB2 ESAF connections
- IMS Connect regions
- IMS - MQ ESAF connections
- IRLM region
- MPP / JMP regions
- OM region
- online logs
- RM region
- SCI region

Additionally, the following IMS low-level resources are monitored:

- DEDB areas and databases
- DLI databases
- HALDB partitions and databases
- Programs
- Transactions

Event Pump monitoring for IMS subsystems processes events and checks the status of resources:

- Event processing traps messages produced by IMS subsystems and related regions using the NetView message automation facility. Based on the effect on IMS subsystems and related resources that are indicated by these messages, events that result in state changes are sent to Netcool/OMNIbus.
- During polling processing, Event Pump periodically checks the status of IMS resources and sends events to Netcool/OMNIbus if exceptions are found, or exceptions no longer exist (resolutions).

Table 34 provides an overview of the tasks that are required to configure IMS to interface with Event Pump.

Table 34. Implementing IMS as an event feed

Task	Where to find information
Make sure that the IMS programs, REXX executable files, messages, and sample members are installed. Verify that the OTMA callable interface is enabled and review security authorizations.	"Installing the Event Pump for IMS components" on page 104

Table 34. Implementing IMS as an event feed (continued)

Task	Where to find information
If you are using large DBCTL regions, install the sample IHS\$MPF exit.	"Installing the IHS\$MPF exit" on page 105
Optional: Install the IMS AO exit (IHSIAOEC).	"Installing the IMS AO exit" on page 106
Configure the program-to-program interface (PPI).	"Configuring the program-to-program interface (PPI)" on page 107
Configure the NetView components.	"Configuring the NetView components" on page 107
Verify the installation.	"Verifying the installation" on page 108
Disable logging of all Event Pump events to the NetView log (update the IHS\$UPRM member).	"Disabling the logging of all Event Pump events to the NetView log" on page 109
Optional: Stop all IMS event processing by using the NetView automation table.	"Customizing event processing" on page 109
Customize the polling operation and thresholds (update the IHS\$UPRM member).	"Customizing polling processing operation and thresholds" on page 110
Exclude resources or events from monitoring.	"Excluding resources and events" on page 112
Optional: Reduce the impact of monitoring large IMS subsystems.	"Tuning for IMS monitoring" on page 113

Installing the Event Pump for IMS components

Verify that the IMS REXX executable files, messages, and sample members are installed. These data sets are installed as part of the SMP/E installation of the product. For installation instructions, see the Event Pump program directory.

Enabling the OTMA callable interface

For the TM/DB and DCCTL IMS regions, the Event Pump uses the OTMA callable interface to issue IMS commands. If either the OTMA callable interface or the OTMA command interface is not enabled, follow these steps:

1. In the IMS startup parameter member (DFSPBxxx) or in the override parameters at IMS start up, specify the following parameters and restart IMS:

```
OTMA=Y
GRNAME=OTMA_XCF_Groupname
OTMANM=IMS_XCF_Membername
```

OTMANM is an optional parameter. If OTMANM is not specified, the IMS APPLID is used.

2. Install and enable the OTMA command interface. For more information, see the *Open Transaction Manager Access Guide and Reference* that is applicable to the IMS version that you are using. Installing and enabling the OTMA command interface typically includes running the DFSYSVI0 OTMA stand-alone program immediately after every z/OS IPL.

The DFSYSVI0 program must run before the monitoring of any IMS region can start and must run before the startup of the Tivoli NetView for z/OS region operating the Event Pump. See the OTMA Callable Interface section in the *Open Transaction Manager Access Guide and Reference* for more information.

Configuring the IMS security parameters

Configure your security software to enable the following access:

- The IMS integration uses the IHSIAUT0, IHSIAUT1, IHSIAUT2, and IHSIAUT3 NetView operators. You must have user IDs with these same names defined to your security software.
- The NetView program issues commands using the OTMA Command Interface. To secure the OTMA Command Interface using the RACF IMSXCF.group.member class member, define a generic IMSXCF.group.IHSI* definition (where *group* is the XCF group name used for OTMA communication) for the IMS integration to have sufficient access.
- Define the following security access to the user ID under which the NetView region runs and to the IHSIAUT0, IHSIAUT1, IHSIAUT2, and IHSIAUT3 user IDs:
 - Access to issue all IMS display commands through the OTMA interface for TM/DB and DCCTL regions, and for the MCS interface for DBCTL regions. To enable the OTMA interface access, set the OTMASE=F (full security) IMS startup parameter and tailor your security product to have access to IMS display commands. If OTMASE=N (none) is specified, the IMS default is to *not* allow OTMA access to IMS display commands. In this case, DFSCCMD0, a user-written command exit, is required to allow this access.
 - Access to allocate temporary data sets.
 - READ access to all IMS ACBLIB, MODBLKS, STEPLIB, and PROCLIB data sets.
 - If you are securing the OTMA interface and defined READ access to the IMSXCF.group.IHSI* RACF profile.
 - CONTROL access to the RECON data sets (to run the DSPURX00 utility).
 - Access to run the DSPURX00 DBRC utility.
- Define the IHSIAUT01 user ID to your security product and require access to TCP/IP FTP. To define the IHSIAUT01 user ID to your security product and require access to TCP/IP FTP, you must have an Open MVS RACF segment or similar segment.
- To submit batch jobs, provide the appropriate access to the user ID under which the NetView region runs and the IHSIAUT01 user ID.

Installing the IHS\$MPF exit

If you are not running IMS DBCTL regions, you can skip this section.

The Event Pump for IMS uses the MCS interface to issue commands to IMS DBCTL regions. Output from these commands is displayed in the NetView for z/OS log, which can flood the log for large DBCTL regions. If you are using large DBCTL regions, you can install the IHS\$MPF sample exit to prevent this output from being displayed in the NetView for z/OS log.

To install the IHS\$MPF exit sample, do the following steps:

1. Review the IHS\$MPF exit sample in the SGTMSAMP library.
2. Modify and submit the IHS\$MPF job to assemble and link edit the exit.
3. Update the MPFLSTxx member as follows:

```
DFS000I,SUP(NO),AUTO(YES),USEREXIT(IHS$MPF)
DFS4444I,SUP(NO),AUTO(YES),USEREXIT(IHS$MPF)
```

See *z/OS MVS Initialization and Tuning Reference* for more information about the SYS1.PARMLIB library.

Usage note: The IHS\$MPF member in the SGTMSAMP library contains detailed instructions about assembling this exit.

Installing the IMS AO exit

Many IMS messages that are required for Event Pump monitoring are issued only to the IMS Master Terminal (MTO). To monitor these messages, Event Pump requires that they be displayed in the z/OS system log. To display IMS messages in the z/OS system log, use an IMS AO type 2 exit (IHSIAOE0) that is installed in every IMS region.

If you already have an IMS AO type 2 exit in use, you can install the optional Event Pump for IMS controller exit (IHSIAOEC). This exit enables up to four other IMS AO type 2 exits to coexist with the IHSIAOE0 exit.

If the IMS controller exit is required, follow these steps to install the IHSIAOEC exit:

1. Determine the order of all IMS AO type 2 exits. The Event Pump IHSIAOE0 exit must be the first exit that is called, followed by other exits in an order determined by the administrator. The IHSIAOE0 exit does not suppress or modify any messages, and the next exit that is called by the IHSIAOEC controller exit sees all the messages that are originally passed to the IHSIAOE0 exit.
2. Determine if the exit reply flag is returned to the IMS program. Only one exit can have its reply value (passed back in the AOE0RPLY field) returned to the IMS program. The IHSIAOE0 exit does not require this field.
3. Update the IHSIAOEE member in the SGTMSAMP library, as described in the member, to indicate the IMS AO exits, their order, and whether their reply value is passed back to the IMS program.
4. To assemble and link edit the IHSIAOEE table, configure and submit the IHSIAOEJ job supplied in the SGTMSAMP library.
5. To assemble and link edit the IHSIAOEC controller exit, configure and submit the IHSIAOCJ job using the instructions provided in the member.
6. Link edit the IHSIAOMT table.
Configure and submit the IHSIAOML job in the SGTMSAMP library. This link-edits the table of IMS messages that are echoed to the z/OS system log by the IHSIAOE0 AO exit. Do not modify this table.
7. Install the IMS IHSIAOE0 AO exit.
Configure and submit the IHSIAO0J job that is in the SGTMSAMP library. If the IHSIAOEC controller exit is used, the ALIAS DFSA0E00 link-edit statement must be removed so that it does not replace the DFSA0E00 alias that points to the IHSIAOEC controller exit.

For additional information about the AO exits, see Appendix E, “Event Pump IMS AO exits,” on page 185.

Verify the AO exit installation

After the Event Pump IMS AO exits are installed, check for the following messages in the system log of the IMS region at startup:

```
IHS363I SUCCESSFULLY LOADED EXIT IHSIAOE0
IHS370I AO EXIT IHSIAOE0 INITIALIZED FOR IMS version type SYSTEM
IHS371I SUCCESSFULLY LOADED MESSAGE TABLE IHSIAOEM
```

Important: The IHS363I message is displayed only if the Event Pump IHSIAOEC AO Controller exit is installed. If you installed the IHSIAOEC exit, the following messages indicate a successful installation:

```
IHS360I AO CONTROLLER EXIT IHSIAOEC INITIALIZED FOR IMS version type SYSTEM
IHS361I SUCCESSFULLY LOADED EXIT CONTROL TABLE IHSIAOEE
```

Determine coexistence issues with CA OPS/MVS

The CA OPS/MVS program has an optional feature called IMS Operations Facility (IOF), which dynamically inserts an AO exit into an IMS region when IMS is started. The AO exits are displayed as OPIMAOEH and perform a controlling function. All references to AO exits are related to Type 2 AO exits only.

The OPIMAOEH AO exit is inserted between IMS and the standard DFSAOE00 AO exit, from where it does the functions that are required by the IOF AO exit, which drives the OPSAOE00 AO exit and then calls the DFSAOE00 AO exit. This enables the coexistence of the IOF AO exit with another AO exit.

If CA OPS/MVS and Event Pump are the only two products that have AO exits that must coexist, then you do not have to install the IHSIAOEC controller exit.

If there are additional AO exits to install other than the AO exits that are supplied by CA OPS/MVS and Event Pump, do the following steps to install the IHSIAOE0 exit to coexist with the IHSIAOEC controller exit:

1. Install the Event Pump IHSIAOE0 exit.
2. Install the IHSIAOEC controller exit. However, modify the IHSIAOEE exit control table so that the IHSIAOEC controller exit is driving the IHSIAOE0 AO exit first, followed by the additional AO exits.
3. Set the flag so that one of the additional AO exits has control over the reply value to be returned to the IMS program, such as in the following example:

```
DC X'00',CL8'IHSIAOE0'
DC X'00',CL8'DFSAOE01'
DC X'80',CL8'DFSAOE99'
```

The OPIMAOEH and OPSAOE00 IOF AO exits do not have to be in the IHSIAOEE exit control table because these exits are run before the IHSIAOEC controller exit starting.

Configuring the program-to-program interface (PPI)

Make sure that the GPZPARMS member contains the following statement:

```
PPI=YES
```

Note: In a 3-address-space environment, review and update the PARMPUMP member.

Configuring the NetView components

The Event Pump for the IMS program runs within a NetView started task. Ensure that all the steps in Chapter 21, "Tivoli NetView for z/OS," on page 123 are completed.

After all customization steps are complete, restart the NetView program. If this restart is the first time the NetView program is started with the installed Event Pump program, all IMS and related regions are automatically detected.

Verifying the installation

After you restart the NetView program, confirm that the installation completed successfully. Do the following steps to verify the installation:

- “Verifying host customization”
- “Verifying end-to-end connectivity”
- “Verifying IMS event monitoring”
- “Verifying poll monitors” on page 109

Verifying host customization

The configuration steps for the host code are tested by the Installation Verification Procedures (IVP) program. To run the IVP program, issue the following command to the NetView program, and correct any errors that are reported:

```
IHS$IVP FEED=IMS NAME= ALL | jobname OUT=CONS|LOG|DSN|APPEND|SCREEN DSN=dsname
```

OUT The following values are for the **OUT** parameter:

CONS Sends the output to the Netcool/OMNIbus EIF probe and the NetView log.

LOG Sends the output only to the NetView log.

DSN Sends the output to the data set.

APPEND
Appends to the data set.

SCREEN
Sends the output to the window. This value is the default value.

DSN The specified data set name. This parameter is required if the **OUT=DSN** or **OUT=APPEND** parameter is specified.

Verifying end-to-end connectivity

After the host IVP program completes successfully, issue the following command to the NetView program to verify end-to-end connectivity:

```
IHS$DEBUG SEND IMSS ims_job_name TEST EVENT  
IHS$DEBUG SEND IMDB ims_job_name TEST EVENT  
IHS$DEBUG SEND IMTM ims_job_name TEST EVENT
```

Verify that the test event is received by the Netcool/OMNIbus EIF probe. Issue the following command to the NetView program:

```
IHS$DEBUG SEND IMSS ims_job_name OK  
IHS$DEBUG SEND IMDB ims_job_name OK  
IHS$DEBUG SEND IMTM ims_job_name OK
```

Verify that the test event is cleared on the Netcool/OMNIbus EIF probe.

Verifying IMS event monitoring

1. Shut down an IMS system or related region. Confirm that the status of that region on the Netcool/OMNIbus EIF probe changes correctly, that is, changes to a critical (red) event.
2. Shut down an IMS dependant region.

3. Confirm the status change on the Netcool/OMNIbus EIF probe.
4. Restart the IMS dependant region. Restarting the IMS dependant region verifies that the NetView automation table is trapping messages correctly.

Verifying poll monitors

1. Stop a program for an IMS region.
2. Wait for the time specified in the IHSI.\$PGMINT parameter in the IHS\$PARM member.
3. Confirm the status change on the Netcool/OMNIbus EIF probe for that program.

Disabling the logging of all Event Pump events to the NetView log

By default, the IHS_LOGGING statement in the IHS\$PRM member is ON:
IHS_LOGGING ON

This results in all Event Pump events being logged to the NetView log.

When installation is complete and verified, create an IHS_LOGGING statement in the IHS\$UPRM member with a value of OFF. Then, refresh this member, by issuing the following command from the NetView command line:

```
IHS$PRMU REFRESH IHS$UPRM
```

Customizing event processing

Messages from the IMS region and related regions are trapped using the Tivoli NetView for z/OS message automation table, including many messages typically sent only to the IMS master terminal (but echoed to the z/OS system log by the Event Pump IMS AO IHSIAOE0 exit, described in “Installing the IMS AO exit” on page 106). These messages are processed and sent as events to the Event Pump database.

The following list describes the steps in event processing:

1. Tivoli NetView for z/OS retrieves messages from the IMS region (and the Event Pump IMS AO exit) and related regions through the message automation table.
2. The Event Pump running under Tivoli NetView for z/OS processes these messages and creates events, as appropriate.
3. The Tivoli NetView for z/OS program-to-program interface (PPI) forwards these events to the source collector through the Event Pump PPI receiver.
4. The PPI receiver formats the events and places them on the event distributor queue.

Events in the Event Pump are categorized by state and priority. The state and priority settings are changed so that only events of a selected significance level are propagated to Netcool/OMNIbus.

You can stop all IMS event processing by running the following Tivoli NetView for z/OS command:

```
AUTOTBL DISABLE NAME=IHSIMAT
```

IMS event processing is enabled again running the following command:

```
AUTOTBL ENABLE NAME=IHSIMAT
```

Polling processing is not affected when you stop event processing.

IMS subsystem up, down and abend events

When an IMS subsystem shuts down, all of its dependent resources are unavailable. During a typical IMS shutdown, the IMS Data Source mainframe component sends a single IMS Subsystem Down event to the Event Pump server. For an abend, a single IMS Subsystem Abend event is sent. The IMS Staged Event Loader component propagates this event to the dependent child resources of the IMS subsystem. For example, IMPG, IMTX, IMDA, IMAR, IMPA, IMLG, IMD2, IMMQ child resources.

For large subsystems, this single event could affect many thousands of child resources. To minimize the use of this processing, events are only propagated to child resources that are in business systems or already have another event assigned to them. For all other related child resources, the resource AlertState is set to critical (red), and the state is set to IMS_Down. You can view these resources through the IMS Resource view, but opening the Properties page does not show any assigned events.

The processing is similar for IMS Subsystem Up event, with the following differences. An IMS Subsystem is started COLD or WARM. For a cold start, all its resources are reset, but for a warm start the resources remain in the same condition that they were before the subsystem was stopped. To reflect this IMS behavior, for a cold start, the IMS Subsystem Up event clears all assigned exceptions. For a warm start, the exceptions are not cleared.

In both warm and cold starts, the normal (green) messages are assigned to child resources that are in business systems, or already have another event assigned to them. For all other related child resources, the resources AlertState is set to normal (green) and the state is set to Available.

Customizing polling processing operation and thresholds

The Event Pump periodically polls the status of IMS resources using these poll monitors:

Database (DBS)

Creates a DBS_%, AREA_%, or PART_% exception event for each database, DEDB area, or HALDB partition that is found to be in an error state.

Transaction (TXN)

Creates a TXN_% exception event for every transaction that is found to be in an error state.

Transaction queue (TXQ)

Creates a TXQ_% exception event when the number of queued transactions is found to exceed the threshold set by you (see “Customizing polling processing operation and thresholds”). This poll monitor also creates a TXQ_% transaction event when the number of input messages for a transaction exceeds a threshold set by you, and there is no MPP/JMP available to process the transaction.

Program (PGM)

Creates a PGM_% exception event for each program that is found to be in an error state.

IMS master terminal (MTO)

Creates an MTO_% exception event if the IMS master terminal is found to be in an error state.

MSC link (MSC)

Creates an MSC_% exception event for each MSC link that is found to be in a restricted status.

You can use these poll monitors to automatically resolve previously active exceptions. Poll monitors send exceptions only the first time that they are detected. If, for example, a program is stopped for several hours, only one exception event is sent (although the poll monitor detects this error every time it runs).

The Event Pump uses a timer to monitor the following events:

IMS log archive time

A DFS2484I event is generated when a log archive job does not finish within the specified time. This event is automatically resolved when the log becomes available (for example, when log archiving completes).

Number of available logs

A DFS3257I event is generated when the number of available logs is below the specified value. This event is automatically resolved when the number of available logs is above the specified value.

To customize the polling process, follow these steps:

1. Review the IHSIPRM member in the SGTMSAMP library (included by the IHS\$PARM member) that contains values for polling monitor thresholds, intervals (for example, how often the polling monitors run), and other customizable values for the poll monitors for all IMS regions.
2. Copy any statements in the IHSIPRM member that you want to change to the IHS\$UPRM member of the SGTMSAMP data set and change the values for your installation.
3. Refresh this member, by issuing the following command from the NetView command line:

```
IHS$PRMU REFRESH IHS$UPRM
```

Refer to the IHSIPRM and IHS\$UPRM members for more information.

Example: Limiting events generated by polling monitors

The number of exception events that are created by a polling monitor might be high for larger subsystems. This high number can affect Event Pump operation. You can avoid high numbers of exceptions by implementing a threshold on the number of exception events that are sent by the poll monitors. The following situation can occur if a poll monitor detects more errors for an IMS database (including DEDB areas and HALDB partitions), programs, or transactions than is specified on the IHSI.\$POLLTHRESH statement in the IHSIPRM member:

- Sends one IHS743W_*type* (where *type* is DATABASE, PROGRAM, or TRANSACTION) exception event
- Stops any other polling processing event for that type from being sent when the IHS743W_*type* exception is active

Stopping any other polling processing event for that type from being sent when the IHS743W_*type* exception is active prevents the source collector from being flooded with many events. Event processing for the particular poll monitor resumes automatically when the number of errors for that resource (DATABASE,

PROGRAM or TRANSACTION) falls below the IHSI.\$POLLTHRESH value and the IHS743W_type exception is automatically resolved.

This processing does not affect the sending of resolution events for existing (and still active) exceptions.

Viewing polling monitor tasks

Each IMS region has its own set of poll monitors, which are effectively Tivoli NetView for z/OS timer commands. To view these timer commands, issue the Tivoli NetView for z/OS LIST command:

```
LIST TIMER=ALL OP=IHSIAUT1
```

Table 35 lists the timers that are displayed.

Table 35. IMS Polling Timers

Command	Timer ID	IMS Subcomponent Checked
IHSIMDBS	DBS x	Databases
IHSIMPGM	PGM x	Programs
IHSIMTXN	TXN x	Transactions
ISHIMMTO	MTO x	IMS Master Terminal
IHSIMTXQ	TXQ x	Transaction Queue Length
IHSIMMSC	MSC x	MSC Link (ESAF)
IHSISUB6	ARC x	Log Archive Job elapsed time

Usage notes:

1. To find out the value of x (the internal Event Pump ID for the region) in the timer ID, display the Tivoli NetView for z/OS global IHSI.mvsname.ims_jobname.SID.
2. Unlike other timed commands, the ARC timer is not run periodically; it runs a set time period only after an IMS log archive job is submitted.
3. If all transaction poll events are excluded from an IMS region (that is, all TXN_ events for an IMS subsystem in the IHS\$EXCL facility) or all transaction resources are excluded (that is, all IMTX resources in the IHS\$EXCL facility), the transaction poll monitor does not run.
4. If all program poll events are excluded from an IMS region (that is, all PGM_ events for a region in the IHS\$EXCL facility) or all program resources are excluded (that is, all IMPG resources in the IHS\$EXCL facility), the program poll monitor does not run.
5. If all database poll events are excluded from an IMS region (that is, all DBS_, AREA_, and PART_ events for a region in the IHS\$EXCL facility) or all database resources are excluded (that is, all IMDA, IMPA, and IMAR resources in the IHS\$EXCL facility), the database poll monitor does not run.
6. If a poll monitor is not required, it might be easier to set the interval for that poll monitor to a value of 0, rather than use the IHS\$EXCL facility.
7. If the entire IMS region is excluded, no poll monitors run.

Excluding resources and events

These are some situations when you might not choose to monitor some IMS resources or events:

- Monitoring of test and development systems is not required. You can exclude these systems from Event Pump monitoring.
- Monitoring of only some critical applications is required (but not others) within an IMS system. You can exclude the appropriate subset of programs, transactions, and databases.
- You do not choose to monitor exception events from the automatic stopping of databases by nightly batch processing. You can exclude these databases at the start of batch processing, and included again at the end of batch processing.
- Only subsystems and resources must be monitored, but not programs, transactions, or databases. You can exclude resources that are not required for monitoring.

To exclude IMS systems or resources:

1. Edit the IHS\$EXCL member of the SGTMSAMP library, using the format that is documented in the member. This member is automatically loaded whenever the Tivoli NetView for z/OS product is started or you request a rule refresh from Netcool/OMNIBus.
2. To change the exclusion rules, issue the following NetView for z/OS command facility command:

```
IHS#EXCL REFRESH member
```

member

This member name is an optional DSIPARM member name holding the exclude rules. The default value is IHS\$EXCL.

Tuning for IMS monitoring

Tivoli NetView for z/OS processing for Event Pump monitoring of the IMS subsystem might be too large for large IMS subsystems. Perform the following actions to reduce the impact:

- Exclude IMS systems or resources that do not require monitoring. See “Excluding resources and events” on page 112.
- Limit the number of individual exclude rules in the DSIPARM IHS\$EXCL member.
- Fully specify resource types in the DSIPARM IHS\$EXCL member. Specifying a wildcard for resource types increases processing times.
- Limit the refreshing of exclude rules and scheduling these outside of peak usage times.
- If poll monitors regularly produce many exceptions, review the IHSI.\$POLLTHRESH function. See “Example: Limiting events generated by polling monitors” on page 111.

Chapter 19. Resource Management Facility (RMF)

Table 44 on page 133 provides an overview of the tasks that are required to configure integration with Resource Management Facility (RMF) monitors.

Table 36. Implementing RMF monitors

Task	Where to find information
Configure RMF to use Distributed Data Server (DDS).	"Configuring RMF"
Update the GPZPARMS member	"Updating the master address space parameters (GPZPARMS)"
Update the registration statements in the REGDATA member	"Updating the registration data file (REGDATA)"

Configuring RMF

Event Pump uses the RMF DDS. Configure RMF to use DDS. For information about how to do this, see the Resource Management Facility library.

Updating the master address space parameters (GPZPARMS)

To update the GPZPARMS member, uncomment the following RMF parameters:

- RMF_SERVER_IP
- RMF_SERVER_PORT
- RMF_SERVER_INTERVAL
- RMF_SERVER_TIMEOUT

Modify any values required for your installation. For additional RMF parameter information see Table 53 on page 155.

Usage notes:

1. You must recycle the master address space (GPZPUMP) for the parameters to take effect. For more information, see "Starting and stopping the master address space" on page 40.
2. In a 3-address-space environment, review and update the RMF startup parameters in the PARMPUMP member in the source collector (GTMPUMP) startup procedure.

Updating the registration data file (REGDATA)

RMF statements define registration that is based on the RMF exception type:

Exception type	Description
DEFAULT	Specifies the default set of RMF exceptions. These include 25 exceptions with production level thresholds. The exceptions are defined in the RMFDEFLT member, which must be in the GTMMON DD statement of the GPZPUMP procedure.
STRESS	Specifies the stress set of RMF exceptions. This exception set has threshold values set to low levels

and is provided for testing exception messages. The exceptions are defined in the RMFSTRES member, which must be in the GTMMON DD statement of the GPZPUMP procedure.

CUSTOMER Indicates a set of exceptions that are user-defined.

Usage notes:

1. In a 3-address-space configuration, the GTMMON DD statement is located in the GTMPUMP procedure.
2. For additional information about the RMF registration statement, see *Event Pump for z/OS Configuration Reference*.

Including RMF statements

Initially, consider using the product-supplied DEFAULT and STRESS exceptions. You can add CUSTOM exceptions at a later time.

Add the following statements to the REGDATA member to enable the RMF event feed:

```
GROUP RMF
RMF TYPE=DEFAULT
RMF TYPE=STRESS
GROUP END
```

Enable the updated REGDATA member using the following z/OS command:

```
f gpzpump,register rmf
```

Customizing RMF exception definitions

If you provide your own exceptions using the CUSTOMER type, your exceptions must be defined in the file that is specified by the CONFIG parameter. The exception definition statements have the following format:

```
TYPE metric_name, parm1, parm2, parm3, parm4, parm5, metric_category, metric_ID,  
    metric_comparison, metric_value, metric_description, RMF_interval,  
    RMF_interval_alignment
```

TYPE

Specifies a statement identifier. The value is ignored by exception processing, and can be specified as any value that is not null. The GPZRMFCU sample configuration file uses a value of CUSTOMER for all exception statements.

metric_name

Specifies a unique identifier for the RMF metric that is tested by this exception. Choose any name, but it must be unique among all the exceptions that are registered, including the exceptions that are registered in the RMFDEFLT and RMFSTRES members.

parm1

If the exception class (the *parm3* value) is SYSPLEX or MVS_IMAGE, set this value to NULL. Otherwise, *parm1* specifies the SMF ID of the z/OS image.

The %name% variable can be used to substitute the name of the z/OS image during exception processing.

parm2

This is the name of the resource, based on the exception class. These resource names can be specified. Any other exception class can be specified with a value of asterisk (*).

Exception class	<i>parm2 value</i>
------------------------	--------------------

SYSplex	Empty string
MVS_IMAGE	The z/OS image name; (you can use %name%).
CHANNEL_PATH	The 2-digit path identifier
LOGICAL_CONTROL	The 4-digit LCU
SSID	The 4-digit SSID
COUPLING_FACILITY	The 8-character coupling facility name
CF_STRUCTURE	The 16-character coupling facility structure name
VOLUME	The 6-character volume identifier

parm3

Indicates the exception class. These exception classes can be specified:

Exception class	<i>parm3 value</i>
SYSplex	Exceptions for the Sysplex
COUPLING_FACILITY	Exceptions for the specified coupling facility
CF_STRUCTURE	Exceptions for the specified CF structure
MVS_IMAGE	Exceptions for the specified z/OS image
I/O_SUBSYSTEM	Exceptions for the I/O subsystem
ALL_CHANNELS	Exceptions that apply to all channels
ALL_SSIDS	Exceptions that apply to all SSIDs
ALL_LCUs	Exceptions that apply to all LCUs
ALL_VOLUMES	Exceptions that apply to all volumes
CHANNEL_PATH	Exceptions for the specified channel
SSID	Exceptions for the specified SSID
LOGICAL_CONTROL_UNIT	Exceptions for the specified LCU
VOLUME	Exceptions for the specified volume
PROCESSOR	Exceptions for the processor
STORAGE	Exceptions for storage
AUXILIARY_STORAGE	Exceptions for auxiliary storage
CENTRAL_STORAGE	Exceptions for central storage
EXPANDED_STORAGE	Exceptions for expanded storage
CSA	Exceptions for the CSA
ECSA	Exceptions for the ECSA
SQA	Exceptions for the SQA
ESQA	Exceptions for the ESQA
ENQUEUE	Exceptions for the enqueue resource
OPERATOR	Exceptions for the operator
SW_SYSTEM	Exceptions for the SW subsystem
JES	Exceptions for JES

HSM	Exceptions for HSM
XCF	Exceptions for XCF

parm4

Indicates the service class period name if the *metric_category* value is P (WLM service class periods). Otherwise, the value is set to NULL.

parm5

Indicates the work scope name, which is based on the *metric_category* value. These values can be specified:

Metric category	<i>parm5 value</i>
G	NULL
J	The job name
P	The period number (1-8)
W	The workload name
S	The service class name
R	The report class name

metric_category

This is an indicator of the work scope type. These values can be specified:

Metric category	<i>parm5 value</i>
G	Indicates the total work scope
J	All jobs
P	WLM service class periods
W	WLM workloads
S	WLM service classes
R	WLM reporting classes

metric_id

Specifies the hexadecimal metric identifier known to RMF. See the RMF documentation for more information about RMF metric identifiers.

metric_comparison

Indicates the comparison operation that is used with the *metric_value* value to compare the actual value of the RMF metric. The following values can be specified:

Operation	Comparison performed
GT	True if actual metric is greater than the <i>metric_value</i> value
LT	True if actual metric is less than the <i>metric_value</i> value
EQ	True if actual metric is equal to the <i>metric_value</i> value
NE	True if actual metric is not equal to the <i>metric_value</i> value

metric_value

The value that is to be compared against the metric.

metric_description

A text description of the metric.

RMF_interval

The interval, specified in seconds, on which to poll RMF for the metric.

RMF_interval_alignment

The number of minutes and seconds past the hour, specified as MMSS, to start *RMF_interval* polling. A value of 0 indicates that polling is running continuously.

The following example shows an exception statement from the GPZRMFCU sample:

```
CUSTOMER HCSANRL, %name%, *, CSA, NULL, NULL, G, 8D0401, GT,  
20,High not released frames %, 100, 0
```

In this example, the type is CUSTOMER and the metric name is HCSANRL. This statement is a CSA metric, so PARM1 is the name of the z/OS image (provided by the %name% variable substitution), the resource name provided by PARM2 is an asterisk, the exception class provided by PARM3 is CSA, and the service class and work scope name provided in PARM4 and PARM5 are both NULL because the METRIC_CATEGORY is G.

The actual metric ID is X'8D0401', which is the CSA High percentage of frames that are not released. The comparison is the number of frames not released that is greater than 20 percent; a percentage of frames not released that is greater than 20 percent generates an exception event. The free form description is "High not released frames %". The interval on which to poll the data is 100 seconds, and the alignment is continuous polling.

Chapter 20. Storage Management Resource Monitoring (SMS)

Event Pump monitors system-managed storage (SMS) resources to generate storage-related messages and to send them to IBM Tivoli Netcool/OMNIBus. Event Pump monitors the following SMS resources:

- Disk Device
- Storage Group
- Storage Volume

Disk devices are represented in the z/OS operating system by a unit control block (UCB) that holds information about an I/O device, such as state information and features of the device. A unique volume serial number is assigned to a disk device during its initialization. A storage group is a group of storage (Storage Volume) resources that can be shared and allocated across multiple applications on z/OS systems, or multiple z/OS images in an SMS complex (SMSplex). Storage Groups are typically shared across all z/OS images that belong to a single SMSplex. An SMSplex is the set of all z/OS images that share the same SMS control data set.

Migration note: The Storage Volume resource corresponds to how the DiskVolume resource was defined in TBSM 3.1. It is renamed according to the IBM common data model.

Table 37 provides an overview of the tasks that are required to configure Event Pump integration with storage management.

Table 37. Implementing Storage Management resource monitoring

Task	Where to find information
Register the SMS as a message source	"Registering SMS as a message source"
Register disk device and storage volume resources	"Registering disk device and storage volume resources" on page 122

Registering SMS as a message source

To set up the JOBLLOG JES SPOOL data sets for use as a message source, add the following statement to the GTMINPUT member that is used for JOBLLOG monitoring:

```
GROUP STORAGE
MONITOR TYPE=STORAGE ID=SMS      *
      CONFIG=ISMSPARM            *
```

The asterisk (*) must be in column 72.

The z/OS messages that begin with IEA, IEC, IEE, IEF, IGD, and IOS are registered as SMS events in the ISMSPARM configuration member. The traps are set for all messages that are listed in the ISMSPARM configuration member.

When messages are trapped by Event Pump, storage-related messages are parsed by the message handler and sent to Netcool/OMNIBus to the resource to which the messages are addressed. The messages are sent by the EIF interface of Event Pump to the queue in the data space. The Event Pump event distributor forwards the events to Netcool/OMNIBus.

Registering disk device and storage volume resources

The Event Pump storage component generates events that are based on interval collection of performance (disk device) or space (disk volume) metrics. These events are created internally by polling both Sysplex data services and DCOLLECT data, and then reporting the exceptions. The disk device and storage volume resources are registered for interval monitoring by using the SMS statement. For information about the SMS statement, see *Event Pump for z/OS Configuration Reference*.

Interval monitoring is enabled or disabled through the registration request. Event Pump receives an indication when the registered metrics exceed the thresholds that you supply during the exception registration process. Exception messages for the affected resource are sent by Event Pump to Netcool/OMNIBus.

This is the interval event flow process:

1. The RMF Sysplex Data Services function returns data to an external calling program on demand. The DCOLLECT program is an IDCAMS service that can be started on demand and returns space usage metrics to the calling program.
2. The RMF and DCOLLECT programs retrieve RMF or DCOLLECT metric data as registered by Event Pump.
3. Event Pump checks the metric data to determine whether a threshold was exceeded. If a threshold was exceeded, Event Pump formats the exception message.
4. Event Pump sends exception messages to the event distributor queue in the data space.
5. The event distributor sends the EIF formatted record to Netcool/OMNIBus.

Chapter 21. Tivoli NetView for z/OS

If you are using Event Pump with an event feed that operates within a NetView for z/OS started task, there are installation and customization steps that are required. The following event feeds use the NetView for z/OS product:

- System Automation for z/OS
- IMS

Usage notes:

1. For NetView for z/OS configuration details for specific event feeds, see the various event feeds.
2. The master and data space address spaces are required, even if an Event Pump event feed runs within the NetView for z/OS address space.

Table 38 provides an overview of the tasks that are required to use the Event Pump with Tivoli NetView for z/OS.

Table 38. Implementing Event Pump with NetView for z/OS

Task	Where to find information
Make sure that the Event Pump REXX executable files, samples, messages, and sample members are installed.	"Installing Event Pump components"
Determine the NetView for z/OS started tasks	"Determine the NetView for z/OS started tasks"
Add the Event Pump libraries to the NetView started task (CNMPROC).	"Updating NetView for z/OS started task JCL" on page 124
Configure the NetView parameter members to enable support for Event Pump.	"Configuring NetView parameter members" on page 124
Review and modify NetView security settings as necessary for your environment.	"Modifying security settings for the NetView program" on page 126
Define the program-to-program interface (PPI) between the NetView started task and Event Pump.	"Defining the program-to-program interface connection" on page 127
Review and modify NetView tuning options.	"Reviewing NetView tuning options" on page 127
Customize the event feed (IMS or Systems Automation)	"Customize event feeds" on page 128

Installing Event Pump components

Verify that the Event Pump REXX executable files, samples, messages, and sample members are installed. These data sets are installed as part of the SMP/E installation of the program. If needed, the installation instructions can be found in the Event Pump program directory.

Determine the NetView for z/OS started tasks

Depending on site configuration, there can be more than one NetView for z/OS started task that is running on each z/OS system, for example, one for network management and one for systems automation.

The Event Pump event feeds that run under NetView for z/OS can all be loaded within a single NetView started task on each z/OS system from which resources are monitored. This task can be an existing NetView started task or one that can be dedicated to Event Pump.

The System Automation for z/OS event feeds for Event Pump must run under the same NetView started task as the System Automation for z/OS program. This restriction exists because the event feed registers as a System Automation Status Observer.

The number of Event Pump address spaces (master and data space) for each z/OS system is not affected by the number of NetView started tasks on that system; one set is required, no matter how many Event Pump event feeds are active on that system.

Customize each NetView for z/OS started task

For every NetView started task that is running the Event Pump, do the following customization tasks:

- “Updating NetView for z/OS started task JCL”
- “Configuring NetView parameter members”
- “Modifying security settings for the NetView program” on page 126
- “Defining the program-to-program interface connection” on page 127
- “Reviewing NetView tuning options” on page 127

Updating NetView for z/OS started task JCL

Add the Event Pump libraries that are shown in the following table to the specified DD concatenations within the NetView for z/OS started task (CNMPROC):

Table 39. Event Pump libraries added to the NetView for z/OS started task

Procedure DD concatenation	Event Pump data set	Notes
STEPLIB	SGTMMODS	Event Pump modules
DSICLD	SGTMEXEC	Event Pump REXX procedures
DSIPARM	SGTMSAMP	Event Pump sample members
DSIMSG	SGTMMSG	Event Pump message files, including the GTMJN01 member that contains Event Pump messages translated into Kanji
CNMPNL1	SGTMPNL1	Event Pump help panels that can be displayed on the NetView console
CNMPNL2	SGTMPNL2	Event Pump help panels translated into Kanji that can be displayed on the NetView console.

Configuring NetView parameter members

Customization for Event Pump consists of two major types:

- Table 40 on page 125 describes the generic changes to the NetView parameter members to enable support for the Event Pump within a NetView started task.

- Table 41 on page 128 and Table 42 on page 128 describe more specific changes to the NetView parameter members to activate (or deactivate) support for individual Event Pump event feeds within a NetView started task.

Some of the customization steps depend on the release of the NetView program that is in use for any particular NetView started task, and on-site conventions and configuration.

The SGTMSAMP library was added to the DSIPARM DD concatenation as shown in Table 39 on page 124. The DSIPARM library is supplied with the NetView for z/OS program.

Table 40. Changes to the NetView parameter members to enable support for the Event Pump within a NetView started task

CxxSTYLE	AUTOCMD.IHS\$MAT.ORDER = statement Make sure that MVSPARM MSGIFAC=CMDONLY is not specified; the NetView program must be able to receive unsolicited messages. To receive unsolicited messages, the task with the CNMCSSIR load module name must be active. Optional. Specify: DEFAULTS AUTOLOGN=YES.
DSIOPFU	%INCLUDE IHS\$OPF in the DSIOPFU member
CNMCMDU	%INCLUDE IHS\$CMD2 in the CNMCMDU member (V5.2 and later)
DSICMDU	%INCLUDE IHS\$CMD in the DSICMDU member (V5.1)
CNMSCAT2	%INCLUDE IHS\$CAT2 in the CNMSCAT2 member
SGTMSAMP Members	
IHS\$EXCL	Review and customize
IHS\$PRM	Review and customize using the IHS\$UPRM sample

Review and configure the following members to define the NetView operation:

- **Automation Table Member** (For NetView Version 5.1 or later): With NetView Version 5.1 or later, the active message automation tables (MAT) are specified in the CNMSTYLE member of the DSIPARM library. You can have more than one MAT entry specified. To define the Event Pump message automation tables to the NetView program, insert the following line in the section on automation tables:

```
AUTOCMD.IHS$MAT.ORDER
```

- **IHS\$MAT** As supplied by the Event Pump, the IHS\$MAT in the SGTMSAMP data set (included in the NetView MAT in previous steps) enables message traps for all Event Pump IMS and System Automation for z/OS event feeds. If you do not use one or more of these event feeds, comment out the corresponding %INCLUDE statement to reduce the Event Pump processing usage.
- **DSICMDU** This is a NetView member that you can customize if you are using Version 5.1. If you do not have a site-customized copy of this member, you can use the sample member that is supplied by the NetView program.

To define the Event Pump commands to the NetView program, insert the following line in the DSICMDU member:

```
%INCLUDE IHS$CMD
```

- **CNMCMDU** This is a NetView member that you can customize if you are using Version 5.2 and later. If you do not have a site-customized copy of this member, you can use the sample member that is supplied by the NetView program.

To define the Event Pump commands to the NetView program, insert the following line in the CNMCMDU member:

```
%INCLUDE IHS$CMD2
```

- **DSIOPFU** This is a NetView member that you can customize. If you do not have a site-customized copy of this member, you can use the sample member that is supplied by the NetView program.

To define the Event Pump auto-operators to the NetView program, insert the following line in the DSIOPFU member:

```
%INCLUDE IHS$OPF
```

- **IHS\$PARM** The IHS\$PARM member includes other members which contain various tuning parameters used by the Event Pump event feeds that run within the NetView program. The IHS\$PARM member is included in the Event Pump SGTMSAMP data set, and is read at NetView startup. Refer to the IHS\$PARM member (and other SGTMSAMP members that it includes) and review each parameter.

The IHS\$DISC_FROM parameter in the IHS\$PRM member allows detected events to flow to Event Pump using the bulk processing path. This threshold protects Event Pump from the flood of typical event traffic that flows through the source collector by detected events. By default, this threshold is set to 5000 but can be set to a lower number if detection event flooding occurs.

The IHS_LOGGING parameter in the IHS\$PRM member is set to ON as the default value. This results in all event records logged to the NetView log for the following event feeds:

- IMS
- Systems Automation for z/OS (V2.3)

The log entries are preceded by the IHS720I message.

When installation and configuration of these event feeds is completed and verified, update the IHS_LOGGING parameter with a value of OFF.

To dynamically set the parameter, issue the following command:

```
IHS$DEBUG LOGGING value
```

where *value* is ON or OFF.

Dynamic changes are not retained across NetView restarts.

You can query the current setting by issuing the following command:

```
IHS$DEBUG LOGGING LIST
```

- **IHS\$EXCL** This SGTMSAMP member contains exclude rules for Event Pump IMS event feeds. If you are using this event feed, review and modify this member as appropriate.

Modifying security settings for the NetView program

Do the following steps to modify the security settings for the NetView program:

1. Define the IHSAUT01 user ID to your security software.
2. To ensure that the IHSAUT01 autotask has sufficient access to issue all NetView commands within the NetView program, do the following steps:
 - If you are using the CNMSCAT2 sample for command security, insert the following line in the CNMSCAT2 sample:

```
%INCLUDE IHS$CAT2
```

- If you are using the CMDAUTH=SAF statement, customize your security product to give the IHSAUT01 autotask access like an operator, with an operator class of 1.
 - If you are using the CMDAUTH=TABLE statement, customize your NetView command authorization table to give the IHSAUT01 autotask access like an operator, with an operator class of 1.
3. Customize the NetView program to enable all autotasks for other Event Pump event feeds that are running under the NetView program the same access to NetView commands as the IHSAUT01 autotask. The names of these autotasks are in the IHS\$OPF member that is included in the SGTMSAMP data set.

Defining the program-to-program interface connection

Define the program-to-program interface (PPI) link between the NetView started task and Event Pump. For configuration details, see the NetView for z/OS library.

Reviewing NetView tuning options

Review the following tuning options when you run the NetView program with the Event Pump event feeds:

- **NetView Started Task Region Size** The detection process for the IMS event feeds can periodically require a large amount of private storage in the NetView address space. The amount of storage required varies, depending on the following factors:
 - The size and number of the subsystems detected.
 - The number of concurrent detections that are occurring.
 - Whether the detections are FULL or PARTIAL.
 - The IHS\$PRM setting on the IHS\$EVTQ_LIMIT parameter. This parameter sets an upper limit on the total number of event and detection records that can be queued in the NetView address space while they are transferred to the Event Pump components. This setting deals with short-term delays in transferring event and detection records to the Event Pump components. For example, the NetView domain that runs the Event Pump code can be started before the Event Pump components are initialized. The default value for this setting is 5000.

If the limit is exceeded, the queue is compressed, and records might be lost; the remaining event and detection records are sent to the Event Pump components by way of the NetView PPI. The IHS745E exception is also sent to the Event Pump database.

The IHS745E, IHS746E, IHS747I and IHS748I messages in the NetView log provide additional indication that limit was exceeded.

If the limit is not exceeded, all the records are sent to the Event Pump components by way of the NetView PPI. The Event Pump data space might then buffer these records while they are transferred to the Event Pump database by way of the network.

The DSI124I and DSI125I messages in the NetView job log might indicate virtual storage shortage problems because of an overly large IHS\$EVTQ_LIMIT value.

The GTM4485I and GTM4486W messages in the Event Pump job log provide an indication that the Event Pump data space is too small to buffer these records while they await transmission by way of the network.

Because processor consumption by Event Pump autotasks can also be significant under the same circumstances, change the REGION size to 0 MB, or to a minimum of 256 MB.

- **IRXANCHR** The IRXANCHR table is a Time Sharing Option Extensions (TSO/E) table that is used to reserve storage for REXX environments. The NetView program and the TSO/E table refer to the IRXANCHR table when allocating storage for each REXX environment that is activated. Event Pump uses the REXX environment extensively. When you start the Event Pump, review the number of REXX environments that are used. See *IBM Tivoli NetView for z/OS Installation: Configuring Additional Components* for information about how to increase the number of environments.

Customize event feeds

The following tables summarize the Event Pump event feed-specific customization steps that are required for the NetView program. For every NetView region that is used, add data sets, and review and update the members as described.

- “IMS event feed”
- “Systems Automation V3.1 event feed”

The SGTMSAMP library was added to the DSIPARM DD concatenation as referred to in Table 39 on page 124. The DSIPARM library is included with the NetView program.

The customization instructions describe how to activate each event feed, typically by ensuring that one or more %INCLUDE statements are run. As supplied by the Event Pump, the %INCLUDE statements for all event feeds are activated. To deactivate any specific event feed, ensure that the relevant %INCLUDE statements are commented out.

IMS event feed

Table 41. Customization steps for IMS event feed

SGTMSAMP Members	Customization instruction
IHS\$MAT	%INCLUDE IHSIMAT
IHS\$OPF	%INCLUDE IHSIMOPR
IHS\$CMD	%INCLUDE IHSICMD
IHS\$EXCL	Review and customize
IHS\$MPF	Review and customize
IHSIDSCV	Optional: Review and customize
IHSIPRM	Review and customize

Systems Automation V3.1 event feed

Table 42. Customization steps for Systems Automation V3.1 event feed

SGTMSAMP Members	Customization instruction
IHS\$MAT	%INCLUDE IHSSMAT
IHS\$OPF	%INCLUDE IHSSOPF
GPZPARMS	Review and customize
IHSSPRM	Review and customize

Chapter 22. Tivoli Workload Scheduler for z/OS

Tivoli Workload Scheduler for z/OS is a program for enterprise-wide production workload scheduling.

Table 43 provides an overview of the tasks that are required to configure Tivoli Workload Scheduler for z/OS to work with Event Pump.

Table 43. Implementing Tivoli Workload Scheduler for z/OS event feed

Task	Where to find information
Update the Event Pump GPZPARMS startup parameter	"Update Event Pump to receive events from Tivoli Workload Scheduler"
Update Tivoli Workload Scheduler to send events to Event Pump	"Update Tivoli Workload Scheduler"

Update Event Pump to receive events from Tivoli Workload Scheduler

Make sure that the following parameters are specified in the GPZPARMS member:

- `OPC_JOBNAME=stcname[,stcname[,...]]`

The stcname= started task name for the Tivoli Workload Scheduler collector/tracker started task name on the system where Event Pump runs. There is one OPC_JOBNAME value for each Tivoli Workload Scheduler task that you want to monitor.

- `EDI_BUFFER_SIZE=2048 | size`

Specifies the size of the external data interface (EDI) buffer within the master address space.

Usage note: In a 3-address-space environment, use the PARMPUMP member instead of the GPZPARMS member.

Update Tivoli Workload Scheduler

Update the Tivoli Workload Scheduler as follows:

1. Include the Event Pump SGTMMODS data set in the STEPLIB concatenation of the Tivoli Workload Scheduler started task procedures.
2. Specify the EXTMON (YES) parameter on the OPCOPTS Tivoli Workload Scheduler initialization statement. This statement defines the runtime options to Tivoli Workload Scheduler for z/OS and the tracker, controller, or the standby controller that uses this statement. Refer to the Tivoli Workload Scheduler for z/OS library for OPCOPTS statement information.
3. To monitor jobs using Event Pump, the external monitor job parameter of an operation must be set to YES (Y). This option is changed in the application program description database and in the current plan. You can use the ISPF panels, or the Tivoli Workload Scheduler for z/OS programming interface to set the option or to browse it.

Part 4. Configuring Event Pump

Chapter 23. Configuring registration data (REGDATA).	133
Update the registration data file	133
Registration data file example	134
 Chapter 24. Filtering Event Pump for z/OS events	 137
Review the GPZPARMS member	137
Message selection and filtering within the Event Pump	137
Event record format	138
Default EIF select table	139
Using the EIF Event logging facilities	139
Starting and stopping the EIF component	140
 Chapter 25. Configuring event-driven registration	 141
Review and update the GPZPUMP startup procedure	141
Update the EDR select table	142
EDR select table example	142
Distribute EDR traps to all Event Pump programs in a sysplex	142
 Chapter 26. Creating user-defined events	 143
Identifying the application to monitor	143
Identifying the messages to monitor	144
Developing the message monitoring logic	145
Developing a handler	146
Predefined variables	147
Predefined fields	147
Creating a message mask member	149
 Chapter 27. Updating message event severity and state using Netcool/OMNIBus	 151
Starting the Netcool/OMNIBus Administrator	151
Working with basic SQL commands	151
Selecting rows in the alerts.zos.severity.msgstate_lookup table	151
Adding new rows to the alerts.zos.severity.msgstate_lookup table	152
Deleting rows from the alerts.zos.severity.msgstate_lookup table	152
Updating rows in the alerts.zos.severity.msgstate_lookup table	152
Using the command-line interface	152

Chapter 23. Configuring registration data (REGDATA)

The customizer is an extension of the Event Pump that enables the processing of new event feeds and provides quick implementation of subsystems or applications. You can define the messages to be monitored and the type of information to be extracted from the messages to use in generating EIF events. A few predefined fields are required and must be included in the event record. These fields can help in developing or changing OMNIbus probe rules.

The customizer enables the use of the generic facilities for processing simple messages and for data source support. You can write a REXX procedure to process messages that require more in-depth parsing or event processing.

After events are created by one of the available customizer facilities, the information from the data source is formatted into an EIF event and sent from the Event Pump to the OMNIbus EIF probe for additional processing.

Table 44 provides an overview of the tasks required to configure the event-driven registration function.

Table 44. Registration data configuration

Task	Where to find information
Ensure that the GPZPUMP startup procedure specifies the registration data file location (GTMINPUT DD statement).	"Updating the GPZPUMP startup procedure" on page 32
Update the registration data file (REGDATA).	• "Update the registration data file" • <i>Event Pump for z/OS Configuration Reference</i>

Update the registration data file

The GTMINPUT DD statement in the GPZPUMP startup procedure specifies the REGDATA member. You define registration statements in this REGDATA member.

Table 45 shows the statements that you can use in a registration data file.

Table 45. Registration data file statements

Statement	Purpose
GLOBAL	Specifies the environment and the system in which the Event Pump source collector component is running.
GROUP	Groups registration data according to user-defined group names.
JOB	Specifies information about the job or started task to be monitored. When used with the MSG operand, defines a generic message trap.
MONITOR	Specifies the type of data sources that are to be monitored.
RMF	Defines registration that is based on the RMF exception type. For more information, see "Updating the registration data file (REGDATA)" on page 115.
SMS	Defines the registration of SMS exceptions that is based on the SMS exceptions type and SMS profile type.

Table 45. Registration data file statements (continued)

Statement	Purpose
SOURCE	Specifies the types of data sources that are used to retrieve state and status information used for monitoring.

Notes:

1. For more information on the registration data file statements, see *Event Pump for z/OS Configuration Reference*.
2. In a 3-address-space configuration, the GTMINPUT DD statement is located in the GTMPUMP procedure.

Registration data file example

Figure 7 contains an example of a registration data file.

```

GLOBAL ENTERPRISE='MY COMPANY'
GLOBAL COMPLEX=RALEIGH
GLOBAL MACHINE=SYSTEM1
GLOBAL LPAR=LPAR1
GLOBAL OS=SYSA
GROUP JOBS
JOB NAME=TESTJOB TYPE=BATC
JOB MSG=IOS001E
GROUP END
MONITOR TYPE=DB2
MONITOR TYPE=TDQ
MONITOR TYPE=CICS JOBNAME=CICS01
MONITOR TYPE=CPSM
GROUP USER_TEST
MONITOR TYPE=TEST *
      DEBUG=YES *
      CONFIG=USERMSG
GROUP JOBLLOG
SOURCE TYPE=JOBLLOG *
      CONFIG=JOBLLOG *
      DEBUG=NO

```

Figure 7. Registration data file example

The GLOBAL statements define the resource hierarchy for objects that are monitored by the remaining registration statements. All events for the TESTJOB job are monitored, as well as any events generated as a result of the IOS001E message. Events that are related to the DB2 and CICS event feeds are also monitored.

The USER_TEST and JOBLLOG groups define custom monitors. Each CONFIG parameter refers to the member of the data set specified by the GTMMON DD statement in the GPZPUMP procedure that defines the custom event source. In Figure 7, the USERMSG member contains statements similar to those in Figure 8.

```

ZEVENT TEST001W
  MASK %S %S # %S* FOR CUSTOMIZER
  EVENTID TEST
  RESOURCE $1
  RESOURCE_TYPE $2
  SEVERITY WARNING
  INFORMATION YES
END

```

Figure 8. USERMSG member example

For more information about defining custom data sources, see Chapter 26, “Creating user-defined events,” on page 143.

The JOBLOG member contains statements similar to those in Figure 9.

```
JOBLOG JOBNAME=TEST0001 DDNAME=JESYSMSG
```

Figure 9. JOBLOG member example

Chapter 24. Filtering Event Pump for z/OS events

Event messages can be intercepted by Event Pump as they are sent to the EIF probe using the Event Integration Facility (EIF). You can filter these z/OS events.

Table 46 provides an overview of the tasks that are required to filter events.

Table 46. Filtering events

Task	Where to find information
Verify that the GPZPUMP startup procedure specifies the location of the EIF select table (GTMEIFSL DD statement).	"Updating the GPZPUMP startup procedure" on page 32
Review the startup parameters.	"Review the GPZPARMS member"
Update the EIF select table (PRMEIFSL member).	"Message selection and filtering within the Event Pump"
Restart the EIF component after updating the EIF select table.	"Starting and stopping the EIF component" on page 140

Review the GPZPARMS member

To start and stop event logging, specify the EIF_LOG and EIF_LOG_LEVEL startup parameters, or use the EIF LOG modify command. If you enable EIF event logging within the master address space, the events are not displayed on the operator console. They are written to the AOPEDLOG data set.

Message selection and filtering within the Event Pump

The GTMEIFSL DD statement in the GPZPUMP procedure defines the EIF select table. By default this is the PRMEIFSL member in the SGTMSAMP data set. The EIF select table is used to select records that are based on the format type, action type, and data within the fields of each Event Pump event record. Each select table rule specifies where the record is sent: Netcool/OMNIBus EIF probe, TBSM, both, or neither.

Review the PRMEIFSL member, and if changes are necessary for your environment, make the updates to a copy of the member in a user-defined data set.

Table 47 shows the statements that you can use in an EIF select table.

Table 47. EIF select table statements

Statement	Purpose
DEFAULT	Sets the default for records not selected by any group SELECT statement.
GROUP	Associates a string of values with a name. You can then use that name within one or more SELECT statements. You can use GROUP statements to maintain a list of values (for example, job names) and to refer to the list multiple times on subsequent SELECT statements.
ONSELECT	Sets the default actions for a SELECT group.
SELECT	Determines where an Event Pump event record is sent. Each SELECT statement starts a new rule. Once selected, the record is not matched against further SELECT statements.

Notes:

1. For more information on the EIF select table statements, see *Event Pump for z/OS Configuration Reference*.
2. In a 3-address-space configuration, the GTMEIFSL DD statement is located in the GTMSRVR procedure.

Event record format

Event Pump event records contain a format, an action, and numerous field number values. The content of each field number varies based on the format code. The format code represents the feed. The action code represents the type of event for the feed. Figure 10 is an example of an Event Pump event record, in this case a CICS exception.

```

FORMAT=19   Type of feed = CICS
ACTION=04   Type of event = Exception
00=2007082714:47:43.  Datetime
01=V310000000067    Sequence Number
02=CITL    Object Type ID = Transaction, local
16=EYMA1PLX    CICSplex Name
17=COEYMA1    CMAS Name
18=CIMM1    CICS Name
24=AOP3    Transaction ID
29=DISABLED    Status
31=TRANSACTION    Resource
32=AOP3    Resource Key
33=MRM    Alert Type
34=VHS    Severity
35=IHSCRDT1    Event Name / ID
36=TRANSACTION NOT ENABLED    Description
66=CONTEXT=EYMA1PLX&SCOPE=
    EYMA1PLX&A_NAME=IHSCRDT1&A_TARGET=CIMM1&A_SEQUENCE=00000001
    &A_PRIORITY=001&A_CONTEXT=EYMA1PLX&A_SEVERITY=VHS

```

Figure 10. Event Pump event record

Event Pump events that match the EIF select table (PRMEIFSL by default) are written to the EIF event listener by the Event Pump, and contain the following slots:

name_value_pairs

Contains all the data from the Event Pump event record in a format that enables the EIF event listener to change the event into the format that is expected by Event Pump.

source

Contains information from the z/OS Communications Server *netname.cpname* field. The Event Pump EIF event listener places the changed Event Pump event on the correct operating system queue within Event Pump.

Figure 11 on page 139 shows an event that is sent to the OMNIbus EIF probe:

```

TBSMV3_SOURCE390;
source=AUIBMQXP.OM02CDRM;
name_value_pairs=
['_FORMAT=19',
'_ACTION=04',
'_F00=2007082714:47:43.979400',
'_F01=V31000000067',
'_F02=CITL',
'_F16=EYMA1PLX',
'_F17=COEYMA1',
'_F18=CIMM1',
'_F24=AOP3',
'_F29=DISABLED',
'_F31=TRANSACTION',
'_F32=AOP3',
'_F33=MRM',
'_F34=VHS',
'_F35=IHSCRDT1',
'_F36=TRANSACTION NOT ENABLED',
'END

```

Figure 11. OMNibus event

The first slot is the event class as defined in the EIF select table for this event.

Default EIF select table

Figure 12 shows the default EIF select table.

```

DEFAULT TBSM=NO TEC=YES
SELECT FORMAT=ALL ACTION=ALL
CLASS='TBSMV3_SOURCE390'

```

Figure 12. EIF select table

The DEFAULT statement sets the destination for the SELECT statements that follow it. In Figure 12, events are sent to the Netcool/OMNibus EIF probe, but not to TBSM. The SELECT statement specifies that all event formats and actions are to be processed, using an event class of TBSMV3_SOURCE390.

The event shown in Figure 10 on page 138 matches the SELECT statement (FORMAT=ALL ACTION=ALL) and is sent to the Netcool/OMNibus EIF probe as shown in Figure 11.

Using the EIF Event logging facilities

Use the EIF event logging facilities to help construct your EIF select table and to debug event problems.

As each SELECT statement is processed, it is internally assigned a sequential number, starting with 1. When logging is enabled, this number is used to identify the SELECT statement that matches an Event Pump event record.

Enable event logging to display the following events:

- The generated event, and return code from the call to the EIF sender
- The Event Pump event record, the default, or SELECT entry number that selected it, and the Event Pump and the Netcool/OMNibus options in effect for the entry that selected the event record
- Both of the above

The level of detail that is logged is controlled by the EIF_LOG_LEVEL startup parameter or the EIF LOG LEVEL modify command.

Use logging to view the following results:

- View the fields and data within each Event Pump event record.
- View the result of the EIF select table matching criteria. You can review the SELECT statements that matched the event record and the options that are in effect for Event Pump and Netcool/OMNIbus.
- View the generated Netcool/OMNIbus event and the return code from the call to the EIF that sent the event to the Netcool/OMNIbus viewer.

In Figure 13, the first SELECT statement matched the data in the Event Pump event record. This event is sent only to the Netcool/OMNIbus viewer.

```
EIF0204 : SELECTED BY SELECT 001 TEC=YES TBSM=NO
EIF0301 : TEC EVENT DATA FOLLOWS
TEC_Generic;source=P390.P390SSCP;name_value_pairs=
'_FORMAT=99','_ACTION=98','_F03=TEST4','_F46=IEF4
03I','_F49=JOB CICS PROD STARTED AT 12:45:59';IDEN
T='TEST4';END
EIF0302 : END OF TEC EVENT DATA
EIF0303 : EIF RC=00000000
```

Figure 13. Sample EIF Log Output

Starting and stopping the EIF component

You can refresh the table dynamically without recycling the master address space. The master address space supports the following modify commands for the EIF component:

EIF STOP

Stops the sending of events to Netcool/OMNIbus. Events are not cached or stored while the interface is stopped. Missed events are lost.

EIF START

Restarts the EIF component after a STOP. The event distributor sends selected records to Netcool/OMNIbus.

EIF REFRESH

Causes the event distributor to read the SELECT table as defined by the GTMEIFSL DD statement and to build a new SELECT table. You can use this command to change the SELECT table without recycling the master address space.

EIF STATUS

Displays information about the status of the EIF component, the connection to Netcool/OMNIbus, and statistics for the records that are sent to Netcool/OMNIbus and Event Pump.

EIF_LOG YES | NO | STATUS

Turns the logging of Event Pump and Netcool/OMNIbus events to the AOPEDLOG file on or off. The default is to display the EIF log status.

EIF_LOG_LEVEL TEC

Controls the level of logging of Event Pump and Netcool/OMNIbus events that are passing through the interface. If the option is missing or not valid, it is ignored and the current log level is displayed.

If you want information about...

Refer to...

EIF commands

Event Pump for z/OS Command Reference

Chapter 25. Configuring event-driven registration

The Event-Driven Registration (EDR) function enables messages to be registered dynamically, based on incoming events.

Dynamic message registration is useful when an incoming message must be paired with a later message, for example, if the incoming message indicates a *status down* event and a later message indicates a *status up* event for the same resource. Event management is controlled by the receipt of a status down event when a resource is unusable, followed by a later status up event when the resource becomes usable again. The status up event clears the status down event and removes it from the views.

In some instances, depending on how the EDI function is implemented, the resource might stay in a down status indefinitely because the status up and status down messages are not sent from the same application. In these cases, the EDR function can be configured to detect that a status down event was sent and to dynamically create a trap for clearing the event. The failed job might also be scheduled to run on another LPAR in the z/OS sysplex. Because of this, create dynamic traps that run for each Event Pump program on the other monitored LPARs within a sysplex.

Incoming events are selected based on statements that are provided in an EDR select table. The statements in the EDR table indicate how to map the incoming events to the appropriate registration record. The clearing event that is provided by default is for the IEF403I started message. You can add additional messages for your enterprise.

Table 48 provides an overview of the tasks that are required to configure the event-driven registration function.

Table 48. Event-driven registration configuration

Task	Where to find information
Review and if necessary update the GPZPUMP startup procedure.	"Review and update the GPZPUMP startup procedure"
Update the EDR select table (PRMEDRSL member).	"Update the EDR select table" on page 142
Distribute EDR traps to all Event Pump programs in a sysplex.	"Distribute EDR traps to all Event Pump programs in a sysplex" on page 142

Review and update the GPZPUMP startup procedure

Review the following DD statements in the GPZPUMP startup procedure and update them as necessary for your installation:

GTMEDRSL

Specifies the member where the EDR select table is located. The PRMEDRSL (GTMSAM25) member contains a sample EDR select table.

GTMDTLDS

Specifies the Dynamic Trap Local Data Store (DTLDS) sequential data set that is used by Event Pump to hold trap creation data that is retained

across Event Pump restarts. Within a sysplex, each Event Pump instance has its own DTLDS data set. The Event Pump dynamic trap resynchronization process automatically reduces the contents of the DTLDS at each Event Pump start up, based on the number of days specified on the DTLDS_MAX_AGE_DAYS startup parameter.

Make sure that the DTLDS data set is allocated with a variable record (VB) format and that it has a minimum record size of 1024 bytes.

Update the EDR select table

Table 49 shows the statements that you can use in an EDR table (PRMEDRSL member).

Table 49. EDR table statements

Statement	Purpose
GROUP	Associates a string of values with a name. You can then use that name within one or more SELECT statements. You can use GROUP statements to maintain a list of values (for example, job names) and to refer to the list multiple times on subsequent SELECT statements.
SELECT	Selects an incoming event record and provides the rules for creating the registration record. After it is selected, the event is not matched against further SELECT statements.

For additional information about EDR table statements, see *Event Pump for z/OS Configuration Reference*.

EDR select table example

If the System Automation for z/OS application sends the status down event, the EDR function creates a trap for the IEF403I message based on the job name that was in the original status down event. When the IEF403I started message is received for the resource in question, a status up event is sent to clear the status down condition for the resource. An example EDR select table follows:

```
GROUP  &JN=GTMDSPC,GTMSRVR,GTMPUMP
SELECT FORMAT=01
ACTION=01
JOBNAME=&JN
JOBTYP=STC
MSG=IEF403I
```

Distribute EDR traps to all Event Pump programs in a sysplex

The Event Pump can be configured to dynamically create traps that are required to monitor for a job to restart and to complete within a sysplex. In this case, the failed job can be scheduled to run on any LPAR within the z/OS sysplex. You can configure Event Pump on every monitored LPAR within a sysplex to be able to create dynamic traps, even though the failed job fails on only one of them.

For information on how to set up sysplex support, see “Updating the GPZPUMP startup procedure” on page 32.

Chapter 26. Creating user-defined events

Use the Event Pump customizer to define user events (specific messages that you want to trap). You can define messages to be monitored and the type of information to be extracted from the messages. A few predefined fields are included in every event record. This standardization of predefined fields assists in the development of OMNIBus probe rules. For a list of the fields, see “Predefined fields” on page 147.

The customizer uses internal facilities of Event Pump to monitor and process the messages based on the criteria that you specify. The generic features of the customizer include message and event feed support. You can also write a REXX script to process messages that require more in-depth parsing or event processing.

After the monitoring event is created by one of the available customizer facilities, the information from the event feed is formatted into an EIF event and sent to the OMNIBus EIF probe for additional processing.

When you identify an application to monitor, you must address the following items to deliver an end-to-end solution:

- Identify an application to monitor and determine the level at which you want to monitor this application.
- Identify a message or messages to monitor.
- Determine the information within the message that can be used as the event ID.
- Maintain a table, the `zos_severity_msgstate_lookup` table, to assign the severity of the message being monitored. This is described in Chapter 27, “Updating message event severity and state using Netcool/OMNIBus,” on page 151.

Table 50 provides an overview of the tasks that are required define user events.

Table 50. Defining user events

Task	Where to find information
Identify the application to monitor.	“Identifying the application to monitor”
Identify the messages to monitor.	“Identifying the messages to monitor” on page 144
Examine the messages to monitor and develop logic to monitor these messages.	“Developing the message monitoring logic” on page 145

Identifying the application to monitor

After you identify a requirement to monitor an application, gather requirements to determine what you want to accomplish by monitoring this application. For example, determine the level at which you want to monitor the application: the subsystem level or the lower resources in this application.

Review the requirements for business service management across the enterprise. When gathering requirements, be sure to consult all those affected, such as operations, help desk, technical users, and executive users.

Identifying the messages to monitor

Identifying the messages to monitor is the first step in setting up Event Pump enablement. The supported event feeds are a combination of IBM and customer best practices.

To develop support for an event feed, examine the messages that are written to the system log (SYSLOG). After identifying the messages to be monitored, determine the information within the message that can be used as the event ID. For example, the event ID can be the message ID.

The Event Pump customizer uses the event ID to assign the severity of the message that is processed. The Event Pump customizer sends the event ID as part of the event to the OMNIBus probe rules. When the probe rules are processed, if the severity was not previously set by the user with the customizer, the customizer sets the severity of the message by using the `zos_severity_msgstate_lookup` table described in Chapter 27, “Updating message event severity and state using Netcool/OMNIBus,” on page 151.

Select the field in the message that you want to use as the resource name and, if applicable, the subresource name. The resource name is commonly mapped to the subsystem name or job name that issues the message. The subresource name is a subset of the resource name.

The monitored messages must be assigned an action type. Depending on the message explanation, possible action types include exception message and resolution message.

The following message is an example of an *exception message*:

```
BB000003E WEBSPHERE FOR Z/OS CONTROL PROCESS string ENDED ABNORMALLY, REASON=nstring
```

Identification

The BB000003E message ID is used as the event ID. The value in the *string* variable identifies the resource for which this message was issued.

Explanation

The WebSphere® for z/OS control process address space indicated by the *string* variable ended unexpectedly. This address space is defined as a server instance in the WebSphere for z/OS Administration user interface.

Event Pump

The customizer processes the criteria in the message mask member that is supplied for this message and extracts the *string* variable that identifies the resource name.

OMNIBus probe rules

When the record is processed by the OMNIBus probe rules, the event ID is assigned a severity. In this message example, the severity is set to indicate a critical severity level.

The following message is an example of a *resolution message*. A resolution message clears all previous messages for a resource regardless of the value that is specified for the event ID in those messages. A resolution exception clears a single previously received exception, based on matching the event ID of the resolution exception to the event ID of the previously received event.

```
BB000019I INITIALIZATION COMPLETE FOR WEBSPHERE FOR Z/OS CONTROL PROCESS string.
```

Identification

This message is a resolution message. The BB000019I message ID is used as the event ID. The value in the *string* variable identifies the resource for which this message was issued.

Explanation

Initialization is complete for the specified WebSphere for z/OS control process address space. This address space is defined as a server instance in the WebSphere for z/OS Administration user interface.

Event Pump

The customizer processes the criteria in the message mask member that is supplied for this message and extracts the *string* variable that is used to identify the resource name.

OMNIBus probe rules

When the record is processed by the OMNIBus probe rules, the event ID is assigned a severity. In this message example, the severity is set to indicate a low severity level (green).

Developing the message monitoring logic

As you review the messages, for each message, determine whether it contains enough information on its own to generate an event, or whether you must add additional message processing logic to create or update fields in an event before generating the event.

In the case where additional processing is required before generating an event, you must develop a handler (see “Developing a handler” on page 146). After developing the handler, create a message mask member (see “Creating a message mask member” on page 149).

Example: Generating an event

Consider the following exception and resolution messages:

Exception message

TEST010E EXCEPTION MESSAGE

Resolution message

TEST020I RESOLUTION MESSAGE

These messages contains sufficient information to generate an exception message event and corresponding resolution message event.

To generate an exception event and resolution event, follow these steps:

- Create a member (USERMSGs) in the GTMMON data set with the following contents:

```
*
* TEST010E EXCEPTION MESSAGE
*
ZEVENT TEST010E
  MASK EXCEPTION MESSAGE
  EVENTID TEST
  SEVERITY CRITICAL
END
*
* TEST020I RESOLUTION MESSAGE
*
ZEVENT TEST020I
```

```

MASK RESOLUTION MESSAGE
EVENTID TEST
SEVERITY CLEAR
END

```

- Enable the message traps by including the following statement in the REGDATA member:
MONITOR TYPE=TEST CONFIG=USERMSG

See Table 51 for the generated event for each message.

Table 51. Generated events

Message	Generated event										
TEST010E EXCEPTION MESSAGE	<pre>TBSMV3_SOURCE390;source=NET1.TVT5104; name_value_pairs=['_FORMAT=30','_ACTION=03', '_F00=2011091607:06:34.360953','_F01=51041633', '_F10=5104','_F11=NET1.TVT5104','_F12=EVMPLEX', '_F13=GTM94PUM','_F14=STC','_F15=TEST010E', '_F16=EXCEPTION MESSAGE', '_F17=TEST010E EXCEPTION MESSAGE', '_F18=TEST','_F19=5','_F21=GTM94PUM','_F22=STC'];END</pre> <table> <tr> <th>Node</th><th>Summary</th><th>Last Occurrence</th><th>Count</th><th>Owner</th></tr> <tr> <td>NET1.TVT5104</td><td>TEST010E EXCEPTION MESSAGE</td><td>9/16/11 11:00:38 PM</td><td>1</td><td>Nobody</td></tr> </table>	Node	Summary	Last Occurrence	Count	Owner	NET1.TVT5104	TEST010E EXCEPTION MESSAGE	9/16/11 11:00:38 PM	1	Nobody
Node	Summary	Last Occurrence	Count	Owner							
NET1.TVT5104	TEST010E EXCEPTION MESSAGE	9/16/11 11:00:38 PM	1	Nobody							

Figure 14. Resulting Netcool/OMNIBus exception event

TEST020I RESOLUTION MESSAGE	<pre>TBSMV3_SOURCE390;source=NET1.TVT5104; name_value_pairs=['_FORMAT=30','_ACTION=05', '_F00=2011091607:09:55.779745','_F01=51041634', '_F10=5104','_F11=NET1.TVT5104','_F12=EVMPLEX', '_F13=GTM94PUM','_F14=STC','_F15=TEST020I', '_F16=RESOLUTION MESSAGE', '_F17=TEST020I RESOLUTION MESSAGE', '_F18=TEST','_F19=0','_F21=GTM94PUM','_F22=STC'];END</pre> <table> <tr> <th>Node</th><th>Summary</th><th>Last Occurrence</th><th>Count</th><th>Owner</th></tr> <tr> <td>NET1.TVT5104</td><td>Resolved: TEST020I RESOLUTION MESSAGE</td><td>9/16/11 11:00:38 PM</td><td>1</td><td>Nobody</td></tr> <tr> <td>NET1.TVT5104</td><td>Resolved: TEST020I RESOLUTION MESSAGE</td><td>9/16/11 11:04:00 PM</td><td>1</td><td>Nobody</td></tr> </table>	Node	Summary	Last Occurrence	Count	Owner	NET1.TVT5104	Resolved: TEST020I RESOLUTION MESSAGE	9/16/11 11:00:38 PM	1	Nobody	NET1.TVT5104	Resolved: TEST020I RESOLUTION MESSAGE	9/16/11 11:04:00 PM	1	Nobody
Node	Summary	Last Occurrence	Count	Owner												
NET1.TVT5104	Resolved: TEST020I RESOLUTION MESSAGE	9/16/11 11:00:38 PM	1	Nobody												
NET1.TVT5104	Resolved: TEST020I RESOLUTION MESSAGE	9/16/11 11:04:00 PM	1	Nobody												

Figure 15. Resulting Netcool/OMNIBus resolution event

Developing a handler

A *handler* processes messages that match the MSGMASK keyword on the MONITOR statement; see *Event Pump for z/OS Configuration Reference*. Handlers can process matched messages that require in-depth parsing. Of the many available routines, handlers focus on the following message parsing and field assignments:

- Creating additional fields that are not included in the predefined set of fields
- Changing any of the predefined set of fields
- Setting up fields

The following stem variables are required to process the message mask members that are provided with the GPZSHDEX and GPZSHDLR samples in the SGTMSAMP data set:

Stem variable name	Usage
--------------------	-------

<i>field.n</i>	Contains the 2-digit field number. If the field number is less than 10, you must include the leading zero, 03.
<i>field_value.n</i>	Contains the value for the field.
<i>usefield.n</i>	Is a Boolean name that determines if the field is used. The stem is set to a default value of true.

After you process the message and are ready to send the event, you can call the CreateRecord procedure. This procedure formats the event and sends it to the EIF interface to be then sent to the EIF probe.

Use the following syntax to call the CreateRecord procedure:

```
return_record = CreateRecord(format_code, action_code, debug)
```

where:

return_record

Contains the formatted record that is sent to Event Pump for processing.

format_code

Indicates the format code to be used for this event feed. Valid values are 30 - 99.

action_code

Indicates the action code to be used for this message. The following values are valid:

- 03** Message event
- 04** Exception event
- 05** Resolution event

debug

Indicates if debug statements are to be written. Valid values are YES and NO.

Predefined variables

The following predefined variables can be accessed as values in the assignment of the FIELD*n* option or used in the COND*n* option:

TRAP_TEXT

Specifies the full message text that was trapped.

TRAP_JOBNAME

Specifies the name of the job that issued the message that is being processed.

TRAP_JOBTYPE

Specifies the type of job that issued the message that is being processed. Examples of job types are STC, JOB, TSU, and BATC.

Predefined fields

The predefined fields are described in Table 52.

Table 52. Predefined fields

Field	Contents	Examples
00	Data and time stamp of event creation	2008101223:05:23.762029
01	Sequence number, a combination of SMF ID and a unique sequence number	SP1138217

Table 52. Predefined fields (continued)

Field	Contents	Examples
08	Extension feed	17
09	Source creator ID	PROFILER
10	SMF ID	SP11
11	NETID.SSCPNAME	USCAC001.CCCDRM3
12	Sysplex name	LPAR400J
13	Name of the job that issues the message	\$DFHSM11
14	Type of job that issues the message	STC
15	Message ID	ARC0037I
16	Message text	DFSMSHSM PROBLEM DETERMINATION OUTPUT DATA ARC0037I (CONT.) SETS SWITCHED, ARCPDOX=HSM.SP11.HSMPDAX, ARC0037I (CONT.) ARCPDOY=HSM.SP11.HSMPDAY
17	Complete message	ARC0037I DFSMSHSM PROBLEM DETERMINATION OUTPUT DATA ARC0037I (CONT.) SETS SWITCHED, ARCPDOX=HSM.SP11.HSMPDAX, ARC0037I (CONT.) ARCPDOY=HSM.SP11.HSMPDAY
18	Event ID The default is the message ID, which can be overwritten.	ARC0037I
19	Event severity This value is event feed specific.	5
21	Resource name The default is the job name, which can be overwritten.	\$DFHSM11
22	Resource type The default is the job type, which can be overwritten.	STC
23	Subresource name This field is optional and can be overwritten.	HSM.SP11.HSMPDAX
24	Subresource type This field is optional and can be overwritten.	ARCPDOX
25	Subresource 2 name This field is optional and can be overwritten.	HSM.SP11.HSMPDAY

Table 52. Predefined fields (continued)

Field	Contents	Examples
26	Subresource 2 type This field is optional and can be overwritten.	ARCPDOY
27	This field is optional and subsystem dependent. For example, it might be the CICSplex name for a CICS subsystem, the data sharing group for a DB2 subsystem, or the IMS group for an IMS subsystem.	DSNDB0A
28	Subsystem plex owning name This field is optional and subsystem dependent. For example, it might be the name of the CMAS owning address space for a CICS subsystem, the name of the data sharing group owning address space for a DB2 subsystem, or the name of the IMS group owning address space for an IMS subsystem.	SYSA

Creating a message mask member

After you develop a handler for messages that require additional processing, the next step is to create a message mask member. Create the message mask member in a data set that is defined in the GTMMON DD concatenation.

Message mask members contain the following statement types:

- General handler statement
Use with messages that require additional processing.
- ZEVEN statements
Use with messages that require no additional processing.

For information about statements that you can use in a message mask member, see *Event Pump for z/OS Configuration Reference*.

The following event feeds have default message mask members:

Event feed	Member
CICSplex System Manager	ICICPARM
DB2	IDB2PARM
SMS	ISMSPARM
BMC MainView for z/OS	MZOSPARM
BMC MainView for DB2	MDB2PARM
BMC MainView for CICS	ISMSPARM

After you create the message mask member, specify the member name using the CONFIG keyword on a MONITOR registration statement in the REGDATA member.

To enable monitoring of the application, issue the following command:

```
F taskname,REGISTER name
```

where *taskname* is the master address space and *name* is the REGDATA member name. For additional information about the REGISTER command, see *Event Pump for z/OS Command Reference*.

Chapter 27. Updating message event severity and state using Netcool/OMNIBus

A message severity and state for a z/OS event is assigned from a lookup table that is added to the Netcool/OMNIBus ObjectServer server. The alerts.zos.severity.msgstate_lookup table is added by the zOSEventSupport_sql file that is provided with Event Pump:

- For Windows systems: zOSEventSupport_sql.bat file
- For UNIX or Linux systems: zOSEventSupport_sql.sh script

You can run this script from the Netcool/OMNIBus Administrator or from the command-line interface.

Starting the Netcool/OMNIBus Administrator

Use the following steps to start the Netcool/OMNIBus Administrator:

1. Depending on your system, take one of the following actions:
 - From the Windows start menu, click **All Programs > Netcool Suite > Administrator**.
 - For a UNIX or Linux system, start the GUI with the nco_config shell script in the \$OMNIHOME/bin directory.
2. Connect to the ObjectServer server that contains your z/OS support tables.
The zos_severity_msgstate_lookup table is created in the alerts database. To view the table structure, click **System > Databases** to display the list of databases that are defined, and then click **alerts** to display all the tables within the alerts database.
3. Click the **Data View** tab of the alerts.zos.severity.msgstate_lookup table to see the current set of rows in the table.

Working with basic SQL commands

The Netcool/OMNIBus Administrator application includes a dialog box that you can use to create and issue SQL statements. You can view the data rows and perform table maintenance such as inserting, updating, and deleting data rows. To open the SQL dialog box, click **System > SQL** on the navigation menu. Use this method to perform SQL operations on the data in the alerts.zos.severity.msgstate_lookup table.

Selecting rows in the alerts.zos.severity.msgstate_lookup table

Use these steps to display all the rows in the zos_severity_msgstate_lookup table:

1. Type the following SQL statement:

```
select * from alerts.zos_severity_msgstate_lookup
```
2. On the toolbar, click **Submit** to run the SQL statement.
3. To find rows that match a certain criteria such as all rows with a similar EventID value, use an SQL command that is similar to the following command:

```
select * from alerts.zos_severity_msgstate_lookup where EventID like 'IEF450I'
```

Adding new rows to the alerts.zos.severity.msgstate_lookup table

To add new rows to the alerts.zos_severity_msgstate_lookup table, type the following SQL statement:

```
insert into alerts.zos_severity_msgstate_lookup (EventID, Severity, MsgState,
  IsMessage) values('^IEF9999E', 5, 'Abended', 1);
```

The Console View output in the bottom half of the dialog box displays the result of action command such as insert, update, and delete.

The following example shows how to use escape with special characters that are contained within the string value for a column:

```
insert into alerts.zos_severity_msgstate_lookup (EventID, Severity, MsgState,
  IsMessage) values('^$HASP999', 3, 'Active', 1);
```

Deleting rows from the alerts.zos.severity.msgstate_lookup table

To delete rows from the alerts.zos_severity_msgstate_lookup table, use the following SQL command:

```
delete from alerts.zos_severity_msgstate_lookup where LookupID = 7443
```

Updating rows in the alerts.zos.severity.msgstate_lookup table

To replace the value on a column in a table row, use the following SQL command:

```
update alerts.zos_severity_msgstate_lookup set Severity = 5 where LookupID = 7442
```

Using the command-line interface

This example uses a plain text file that contains all the SQL commands to add three new rows to the alerts.zos_severity_msgstate_lookup table. The file is processed with the isql command-line utility that is provided by Netcool/OMNIbus. The following steps are for Windows systems. Use the same sequence of steps for UNIX systems using the nco_sql command-line utility and follow its usage syntax.

1. Create a plain text file with insert statement for the additional severity lookup rows. The file contents are similar to the following example:

```
insert into alerts.zos_severity_msgstate_lookup (EventID, Severity, MsgState,
  IsMessage) values('MY_ERROR', 5, 'Error', 0);
insert into alerts.zos_severity_msgstate_lookup (EventID, Severity, MsgState,
  IsMessage) values('MY_WARNING', 3, 'Warning', 0);
insert into alerts.zos_severity_msgstate_lookup (EventID, Severity, MsgState,
  IsMessage) values('MY_INFO', 0, 'Informational', 0);
go
```

The **go** statement must be included at the end of the file and the semicolon (;) must be placed at the end of each statement. For UNIX systems, save the file in UNIX format or use a UNIX utility such as dos2unix before using the file.

2. Run the SQL statement with the isql command-line utility:

```
%NCHOME%\bin\redist\isql -S NCOMS -U root
-P "" -i my_zos_severity
```

The command-line output resembles the following example:

```
%NCHOME%\bin\redist\isql -S NCOMS -U root
-P "" -i my_zos_severity_msgstate_lookup_rows.sql
(3 rows affected)
```

Part 5. Appendixes

Appendix A. Startup parameters (GPZPARMS)

The GPZPARMS member contains the startup parameters for the master GPZPUMP startup procedure. Most startup parameters are optional and have default settings. Table 53 lists some startup parameters by function. To review all the startup parameters that you can specify, see *Event Pump for z/OS Configuration Reference*.

Table 53. Startup parameters

Function	Parameter	Purpose
Automation	AFOPER_JOBNAME	Specifies the started task name of the AF/OPERATOR task.
	AUTOOPER_JOBNAME	Specifies the started task name of the AutoOPERATOR task.
	EDI_BUFFER_SIZE	Specifies the EDI buffer size. A value of 5000 is required for the Event Pump automation interface implementation.
	OPSMVS_JOBNAME	Specifies the OPS/MVS main task and server started task names.
	PPI	Specifies whether to use the NetView for z/OS program-to-program interface (PPI). A value of YES is required for the Event Pump automation interface implementation.
	SA390_JOBNAME	Specifies the started task name of the NetView for z/OS task under which NetView for z/OS automation and System Automation for z/OS run.
EIF	EIF_LOG	Turns the logging of Event Pump and Netcool/OMNIBus events on or off.
	EIF_LOG_LEVEL	Controls the level of detail that is logged. Logged data is written to the AOPEDLOG DD file. For additional information, see “Review the AOPEDLOG log” on page 40.
IMS	PPI	Specifies the program-to-program interface (PPI).

Table 53. Startup parameters (continued)

Function	Parameter	Purpose
RMF	RMF_SERVER_IP	Supplies the IP address of the Distributed Data Server (DDServer) address space. This value is the IP address of the system on which the RMF DDServe resides. If the IP address of the system on which the RMF DDServe resides is not provided, the default value of the local MVS systems IP address is used. The local MVS systems IP address can be used when a single instance of RMF DDServe is collecting data for multiple systems within a sysplex. If the DDServe is not in the same logical partition (LPAR) as the source collector, set the RMF_SERVER_IP startup parameter before the source collector is started.
	RMF_SERVER_PORT	Specifies the port number on which RMF responds. This value corresponds to the DM_PORT parameter in the GPMSRV00 member of the IEFARM library.
	RMF_SERVER_INTERVAL	Corresponds to the MINTIME option for RMF Monitor.
	RMF_SERVER_TIMEOUT	Specifies the value in seconds that the source collector waits before assuming that the Distributed Data server is unavailable. If a request times out, all future collections are not attempted until at least the time that is indicated by the server interval value has passed.

Table 53. Startup parameters (continued)

Function	Parameter	Purpose
Sysplex support	BACKLOG_SIZE	Specifies the Event Pump HTTP server maximum number of queued requests.
	DTLDS_MAX_AGE_DAYS	Specifies the number of days to keep dynamic traps before pruning them.
	GETTRAPS_USERS	Specifies the list of valid user IDs that are allowed to call the SOAP GETTRAPS function.
	LISTEN_TIMEOUT	Specifies the Event Pump HTTP server listen timeout value.
	LOG_REQUESTS	Indicates whether the incoming requests to Event Pump HTTP server are logged to the AOPOUT file.
	NOTIFY_USERS	Specifies the list of valid user IDs that are allowed to call the SOAP NOTIFY function.
	PORT	Specifies the TCP/IP port number on which the Event Pump HTTP server is to listen.
	REGISTER_USERS	Specifies the list of valid user IDs that are allowed to call the SOAP REGISTER function.
	<i>smfid</i> _USER	Specifies the user ID to be used when sending SOAP requests to the partner Event Pump that is identified by the <i>smfid</i> parameter.
	SOCKETSET_SIZE	Specifies the Event Pump HTTP server default socket set size.
	SYSPLEX_SUPPORT	Specifies whether sysplex support is enabled.
SOAP interface	VALIDATE_SOAP_USERS	Specifies whether SOAP requests require a valid user ID and password.
Tivoli Workload Scheduler	OPC_JOBNAME	Specifies one or more Tivoli Workload Scheduler started tasks.
	EDI_BUFFER_SIZE	Specifies the EDI buffer size.

Usage notes:

1. You must recycle the master address space (GPZPUMP) for the parameters to take effect. For more information, see “Starting and stopping the master address space” on page 40.
2. In a 3-address-space environment, review and update the startup parameters in the PARMPUMP member in the source collector (GTMPUMP) startup procedure.

Appendix B. Event Pump samples reference

Table 54 lists the samples that can be found in the SGTMSAMP data set. The table includes a brief description of each sample.

The samples are tested and are supported. You are responsible for making modifications to the samples for your environment.

Table 54. List of Samples

Distributed As	Name	Description	Used by
DEFLOGS	same	Allocates Event Pump log data sets	
GPZ\$EXCL	same	Exclude filter definitions for DB2 and CICSplex System Manager	GTMMON
GPZCDB2C	same	Sample CICSplex SP file to run under the batch definition interface to set up DB2 connection monitoring	
GPZCFILE	same	Sample CICSplex SP file to run under the batch definition interface to set up file monitoring	
GPZCGRP	same	Sample CICSplex SP file to run under the batch definition interface to set up PERIODDEFS and CICS groups monitoring	
GPZCMAS	same	Sample CICSplex SP file to run under the batch definition interface to set up CICS regions and connections monitoring	
GPZCPEDE	CPEDEFLT	Sample monitor file for SMS cache performance exceptions (default)	GPZPUMP, GTMPUMP
GPZCPEST	CPESTRES	Sample monitor file for SMS cache performance exceptions (stress)	GPZPUMP, GTMPUMP
GPZDPEDE	DPEDEFLT	Sample monitor file for SMS device performance exceptions (default)	GPZPUMP, GTMPUMP
GPZDPEST	DPESTRES	Sample monitor file for SMS device performance exceptions (stress)	GPZPUMP, GTMPUMP
GPZDSEDE	DSEDEFLT	Sample monitor file for SMS space usage exceptions (default)	GPZPUMP, GTMPUMP
GPZDSEST	DSESTRES	Sample monitor file for SMS space usage exceptions (stress)	GPZPUMP, GTMPUMP
GPZMSMS	ISMSPARM	Sample monitor file for SMS messages	GPZPUMP, GTMPUMP
GPZCTRAN	same	Sample CICSplex SP file to run under the batch definition interface to set up transaction definitions	

Table 54. List of Samples (continued)

Distributed As	Name	Description	Used by
GPZMCPSM	ICICPARM	Sample monitor file for CICSplex System Manager	GPZPUMP, GTMPUMP
GPZMDB2	IDB2PARM	Sample monitor file for DB2	GPZPUMP, GTMPUMP
GPZMHCK	same	Sample configuration member for Health Checker for z/OS	GPZPUMP, GTMPUMP
GPZMMCIC	MCICPARM	Sample monitor file for BMC MainView for CICS	GPZPUMP, GTMPUMP
GPZMMDDB2	MDB2PARM	Sample monitor file for BMC MainView for DB2	GPZPUMP, GTMPUMP
GPZMMZOS	MZOSPARM	Sample monitor file for BMC MainView for z/OS	GPZPUMP, GTMPUMP
GPZSMS	ISMSPARM	Sample monitor file for SMS messages	GPZPUMP, GTMPUMP
GPZPARMS	same	Source collector startup parameters	GPZPUMP
GPZPUMP	same	Sample procedure for the master address space	
GPZRMFCU	None	Sample monitor file for RMF customized exceptions	GPZPUMP, GTMPUMP
GPZRMFDE	RMFDEFLT	Sample monitor file for RMF default exceptions	GPZPUMP, GTMPUMP
GPZRMFST	RMFSTRES	Sample monitor file for RMF stress exceptions	GPZPUMP, GTMPUMP
GPZSHDEX		Profile handler that processes message mask members	
GPZSHDLR		Profile handler for user monitor entries	
GTMEDITB	same	EDI table member that defines the field numbers for each event record type (FORMAT) and subtype (ACTION)	
GTMSAM02	AOPCICS2	Collects CICS v2 transient data messages	
GTMSAM03	AOPCICS3	Collects CICS v3.1 and later transient data messages	
GTMSAM07	GPZDSPC, GTMDSPC	Data space startup procedure	
GTMSAM08	GTMPUMP	Source collector startup procedure (3-address-space configuration)	
GTMSAM09	GTMSRVR	Event distributor startup procedure	
GTMSAM10	PARMDSPC	Data space startup parameters	GPZDSPC, GTMDSPC
GTMSAM12	PARMPUMP	Event distributor address space parameters	GTMPUMP
GTMSAM22	REGDATA	Registration records	GPZPUMP, GTMPUMP

Table 54. List of Samples (continued)

Distributed As	Name	Description	Used by
GTMSAM23	PRMEIFSL	EIF select table statements	GPZPUMP, GTMSRVR
GTMSAM24	PARMEIF	EIF configuration statements	GPZPUMP, GTMSRVR
GTMSAM25	PRMEDRSL	Event- driven registration (EDR) statements	GPZPUMP, GTMSRVR
ICICPARM	same	Handler message mask that is used by the CICSplex System Management event feed	GPZPUMP, GTMPUMP
IDB2PARM	same	Handler message mask that is used by the DB2 event feed	GPZPUMP, GTMPUMP
IHS\$CAT2	same	Command authorization table statements in the CNMSCAT2 member	NetView program
IHS\$CMD	same	Command definition statements in the DSICMDU member (NetView v5.1)	NetView program
IHS\$CMD2	same	Command definition statements in the CNMCMDU member (NetView v5.2 and later)	NetView program
IHS\$DISC	same	Discovery event file	
IHS\$EXCL	same	Exclude filter definitions for the IMS event feed	
IHS\$OPF	same	NetView operator definitions in the DISOPF member	NetView program
IHS\$MAT	same	NetView message automation table (MAT) entries	NetView program
IHS\$MPF	same	Sample MPF exit to suppress messages from SYSLOG for console names prefixed with IHS, for example the EMCS consoles used to communicate with IMS DBCTL regions.	
IHS\$PRM	same	Common global variables. Do not modify this sample.	
IHS\$UPRM	same	User changes to the common global variables in the IHS\$PRM sample	
IHSIAOEE	same	IMS AO exit control table for use by AO controller exit (IHSIAOEC)	
IHSIAOEJ	same	Assembles and link-edits AO Exit control table (IHSIAOEE) for use by the AO controller exit (IHSIAOEC).	
IHSIAO0J	same	Link-edit IMS AO Exit (IHSIAOE0)	
IHSSMONI	same	Sample for customizing the resources being monitored by System Automation for z/OS	IHSSSTRT

Table 54. List of Samples (continued)

Distributed As	Name	Description	Used by
IHSIPRM	same	Sample that contains values for polling monitor thresholds, intervals, and other customizable values for the poll monitors for all IMS regions Do not modify this sample.	
IHS\$EXCL	same	Exclude filter definitions for IMS and DB2 data sources	
IHS\$PRM	same	Sample for common global variables	
IHS\$UPRM	same	User changes for the IHS\$PRM common global variables sample member	

Appendix C. Configuring the startup procedures using three address spaces

You can configure Event Pump using three address spaces:

- Source collector
- Event distributor
- Data space

Usage note: Consider configuring Event Pump using two address spaces: master and data space. The master address space merges the source collector and event distributor address spaces. By using two address spaces, you shorten the path length and eliminate cross address space communication. For information about the 2-address-space configuration, see Chapter 8, “Getting ready to start the Event Pump program,” on page 31. The 2-and 3-address-space configurations are compatible.

Before configuring the Event Pump procedures, copy the following startup procedures for the address spaces from the SGTMSAMP library to a user data set:

GTMPUMP The source collector startup procedure
GTMSRVR The event distributor startup procedure
GTMDSPC (GTMSAM07)
 The data space startup procedure

Make modifications to the procedures in the user data set. By using the user data set, you protect your modifications when maintenance is applied.

Each startup procedure uses one or more startup parameter members. Copy the following startup parameter members from the SGTMSAMP library to a user data set:

PARMPUMP (GTMSAM08)
 Source collector startup parameter member
PARMSRVR (GTMSAM13)
 Event distributor startup parameter member
PARMDSPC (GTMSAM10)
 Data space startup parameter member
PARMEIF (GTMSAM24)
 EIF configuration file
PRMEDRSL (GTMSAM25)
 Event driven registration (EDR) event file
PRMEIFSL (GTMSAM23)
 EIF select table
REGDATA (GTMSAM22)
 Registration data statements

Figure 16 shows the primary input and output data files that are used by the source collector and event distributor address spaces.

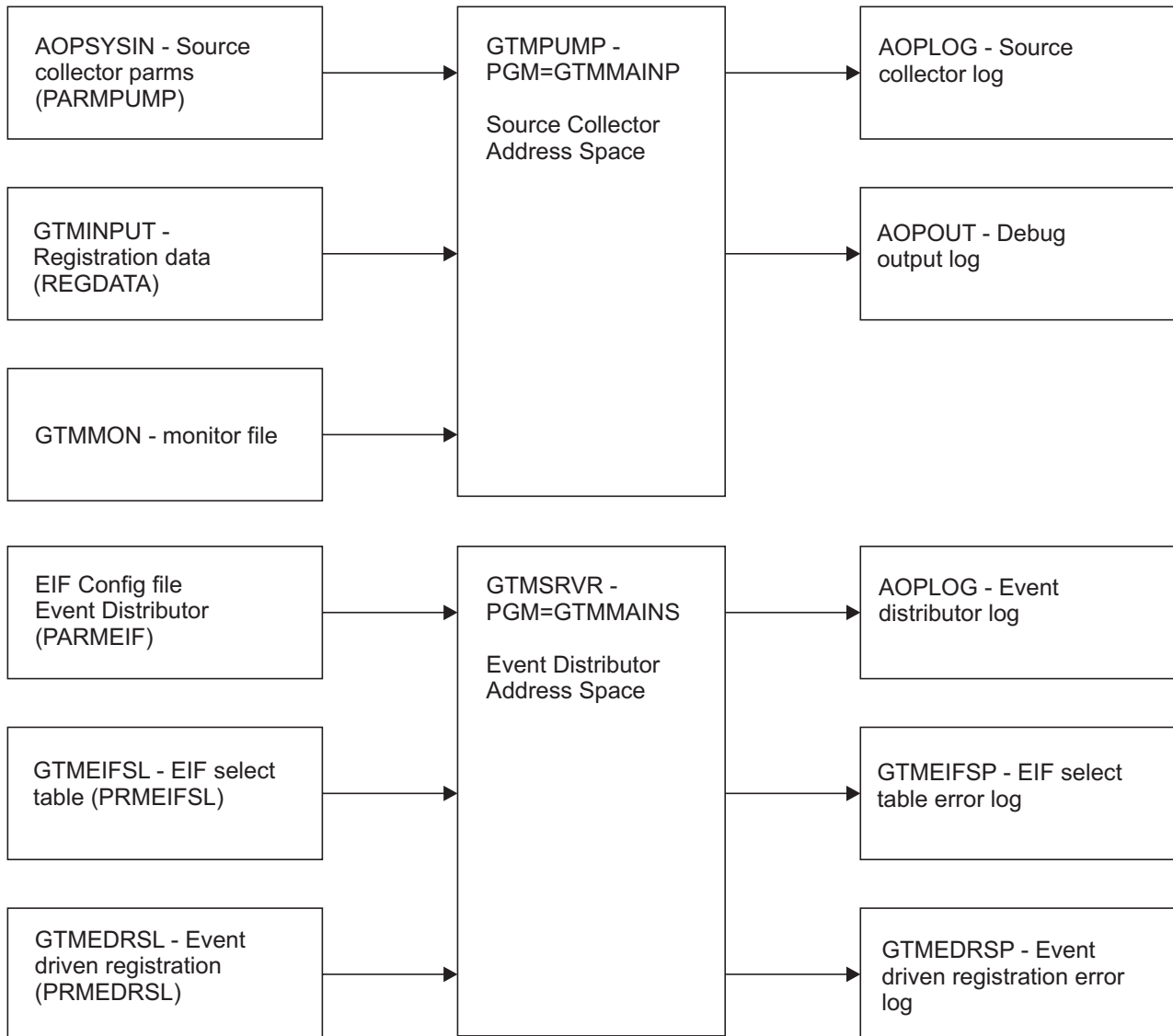


Figure 16. Source collector and event distributor address space data files

Updating the GTMPUMP startup procedure

Update the source collector startup procedure member (GTMPUMP) in the user data set as required for your environment:

1. Update the following high-level qualifiers and associated data set names:
 - Tivoli Event Pump data sets (%hlq)
 - REXX alternate runtime library (%rexshlq)
 - CICSplex System Manager authorized library (%cpsmllib)
2. Change the data set name of the SGTMSAMP data set to specify the data set to which you copied the Event Pump startup parameter members.
3. Optional: If you plan to use Event Pump to monitor CICSplex/SM managed CICSplexes, uncomment the CPSM authorized library :

```
//STEPLIB DD DISP=SHR,DSN=%h1q.SGTMMODS <== AUTH LIB
// DD DISP=SHR,DSN=%rexh1q.SEAGALT <== AUTH LIB
//* DD DISP=SHR,DSN=%cpsmlib.SEYUAUTH <== AUTH LIB
```

If you plan to use the Event Integration Facility (EIF) for filtering, and the SYS1SCREERUN library is not available through your link list concatenation, you must add this library to the STEPLIB DD concatenation. In this case, the library must be APF-authorized.

4. The AOPSYSIN DD statement specifies the PARMPUMP member that contains the startup parameters for the source collector startup procedure:

```
//AOPSYSIN DD DISP=SHR,DSN=%h1q.SGTMSAMP(&MBR)
```

Most startup parameters are optional and have default settings. Review and update the following required parameters:

- COMMAND_PREFIX
- CONSOLE
- SUBSYSTEM_NAME
- TCPIP_JOBNAME, if required by your installation

Additionally, review and update the parameters described in Appendix A, “Startup parameters (GPZPARMS),” on page 155.

For information about the startup parameters, see *Event Pump for z/OS Configuration Reference*.

5. Registration data statements indicate which events Event Pump gathers and forwards. The GTMINPUT DD statement defines the REGDATA member that contains the data registration parameters that are used by Event Pump:

```
//GTMINPUT DD DISP=SHR,DSN=%h1q.SGTMSAMP(&REGMBR)
```

Review the GLOBAL statements in the REGDATA member and customize them as needed. For information about the registration data statements, see Chapter 23, “Configuring registration data (REGDATA),” on page 133.

6. Optional: You can uncomment the GTMMON statement at a later time if you plan to use the Event Pump customizer to process incoming data source messages.

```
//*GTMMON DD DISP=SHR,DSN=%h1q.GTMMON
```

The GTMMON DD statement identifies the monitor file that implements the definitions necessary to monitor messages that are produced by the subsystem. For information on how to define messages, see Chapter 26, “Creating user-defined events,” on page 143.

Usage note: For the CICSplex System Management source, make sure that the GTMMON DD statement references the data set where the ICICPARM member is located. By default, the ICICPARM member is in the SGTMSAMP data set.

7. Optional: The Dynamic Trap Local Data Store (DTLDS) is a sequential data set that holds trap creation data that must be retained across Event Pump restarts. Each Event Pump instance has its own DTLDS data set. Uncomment the GTMDTLDS DD statement if you plan to enable sysplex support:

```
//*GTMDTLDS DD DSN=%h1q.GTMDTLDS,DISP=MOD,
//* DCB=(LRECL=1024,BLKSIZE=0,RECFM=VB),
//* UNIT=SYSDA
```

8. If TCP/IP domain name resolution is not fully configured at your site, uncomment the SYSTCPD DD statement and update the TCPDATA library name:

```
//*SYSTCPD DD DISP=SHR,DSN=%tcph1q.SEZAINST(TCPDATA)
```

9. If you plan to use multiple instances of Event Pump, each instance and its associated data space must include the following statement:

```
//ACC1IDxx DD DUMMY
```

where *xx* is a 2-character suffix that defines the instance. If you omit this DD statement, the ID used by Event Pump defaults to 01.

10. Copy the procedure to your PROCLIB data set.

For additional information about modifying the GTMPUMP procedure, see the procedure prolog.

Updating the GTMSRVR startup procedure

Update the event distributor startup procedure member (GTMSRVR) in the user data set as required for your environment:

1. Update the Tivoli Event Pump data sets high-level qualifiers (%hlq).
2. Change the data set name of the SGTMSAMP data set to specify the data set to which you copied the Event Pump startup parameter members.
3. The AOPSYSIN DD statement specifies the PARMSRVR member that contains the startup parameters for the event distributor startup procedure:

```
//AOPSYSIN DD DISP=SHR,DSN=%hlq.SGTMSAMP(&MBR)
```

Most startup parameters are optional and have default settings. Review and update the following required parameters:

- COMMAND_PREFIX
- CONSOLE
- EIF_CONF (This statement can refer to an HFS file or a Partitioned Data Set (PDS) member.)
- LOG1 (Use the data set names that you specified when you ran the DEFLOGS job.)
- LOG2 (Use the data set names that you specified when you ran the DEFLOGS job.)
- SUBSYSTEM_NAME
- TCPIP_JOBNAME, if required by your installation

For more information on these parameters, see *Event Pump for z/OS Configuration Reference*.

Also review and update the parameters that are described in Appendix A, "Startup parameters (GPZPARMS)," on page 155.

4. If your installation runs more than one TCP/IP stack, the Event Integration Facility (EIF) component must gain affinity to the proper stack using one of the following methods:
 - The BPXTCAFF program can be used to bind affinity to a stack of your choice. The BPTXTCAFF program must run before the program that initializes the address space or before the program that initializes the address space containing the Event Integration Facility.

Depending on your installation, run the BPTXTCAFF program before the event distributor address space. One way of doing this is to add this statement in the event distributor startup JCL before the statement used to start the Event Integration Facility:

```
//STEP0 EXEC PGM=BPXTCAFF,PARM=MYSTACK
```

where the PARM parameter is a pointer to the stack to which you want to establish affinity.

This example shows how the startup JCL might look like after the step is added:

```
//STEP0 EXEC PGM=BPXTCAFF,PARM=MYSTACK
//SRVR EXEC PGM=GTMMAINS
//STEPLIB DD DISP=SHR,DSN=%h1q.SGTMMODS
```

- To request transport affinity when multiple TCP/IP images are used, set the _BPXK_SETIBMOPT_TRANSPORT environment variable to the name of the preferred transport before starting the Event Integration Facility (GTMSRVR started procedures).

For additional information about the BPXTCAFF program and BPXK_SETIBMOPT_TRANSPORT environment variable, see *UNIX System Services Planning*.

5. Event messages can be intercepted within the event distributor as they are sent to the EIF probe using EIF. You can filter these z/OS events by using an EIF select table. The EIF select table name (PRMEIFSL by default) is specified by the GTMEIFSL DD statement:

```
//GTMEIFSL DD DISP=SHR,DSN=%h1q.SGTMSAMP(&EIFSLMBR)
```

The PRMEIFSL member in the SGTMSAMP data set contains sample EIF table statements.

The GTMEIFSP DD statement is used to report any errors with SELECT statements in the PRMEIFSL member:

```
//GTMEIFSP DD SYSOUT=X
```

To set up message selection and filtering, review the contents of the PRMEIFSL member and change it as necessary for your installation. For information about specifying the EIF select table, see Chapter 24, “Filtering Event Pump for z/OS events,” on page 137.

6. Optional: You can use the Event-Driven Registration (EDR) function to dynamically register messages based on other messages that are currently registered. You can define these messages within an EDR event file. To use the event-driven registration function, follow these steps:
 - Uncomment the following GTMEDRSL and GTMEDRSP DD statements:

```
//*GTMEDRSL DD DISP=SHR,DSN=%h1q.SGTMSAMP(&EDRSLMBR)
//*GTMEDRSP DD SYSOUT=X
```
 - Modify the PRMEDRSL member as required for your installation. For information about how to update this member, see Chapter 25, “Configuring event-driven registration,” on page 141.
7. If you plan to use multiple instances of Event Pump, each instance and its associated data space must include the following statement:

```
//ACC1IDxx DD DUMMY
```

where *xx* is a 2-character suffix that defines the instance. If you omit this DD statement, the ID that is used by Event Pump defaults to 01.

8. Copy the procedure to your PROCLIB data set.

For additional information about modifying the GTMSRVR procedure, see the procedure prolog.

Update the EIF configuration file

The name of the EIF configuration file is specified by the EIF_CONF statement in the PARMSRVR member of the SGTMSAMP data set. By default, the EIF configuration file is the PARMEIF member.

To update the EIF configuration file, modify the statements in the PARMEIF member as required for your installation. The ServerLocation and ServerPort statements are required. The other statements are optional and have default settings if not specified.

If you want information about...	Refer to...
EIF configuration file statements	<i>Event Pump for z/OS Configuration Reference</i>

Updating the GTMDSPC startup procedure

Update the GTMDSPC data space startup procedure member in the user data set as required for your environment:

1. Update the high-level qualifier (%hlq) for the Event Pump data sets:

```
//STEPLIB DD DISP=SHR,DSN=%hlq.SGTMMODS  
//ACC1PARM DD DISP=SHR,DSN=%hlq.SGTMSAMP
```
2. Change the data set name of the SGTMSAMP data set to specify the data set to which you copied the Event Pump startup parameter members.
3. The EXEC statement specifies the PARMDSPC member which contains the parameters for the data space startup procedure:

```
//DSPC EXEC PGM=GTMACC17,PARM=&MBR
```

Usage notes:

- a. The LOGQUEUESIZE parameter specifies the size of the log queue in pages. The queue is used to hold data that is written to the log data sets. You can customize the LOGQUEUESIZE startup parameter.
- b. The Event Pump data space holds data in queues as the data is sent between the various Event Pump components. Because of the specific nature of this address space, few problems require your attention. The queues are intended to hold data only for a short time. However, if communication between the host and the event distributor is lost, the event distributor queue might be required to hold events.

The size of the event distributor is determined by the OSQUEUESIZE parameter to the Event Pump data space. The value that is specified is the number of pages dedicated to hold queue pointers. Each page can store 1024 records in the database, not including the first page. The first page also contains queue header information, therefore the total number is less. The size of the data space that is obtained is determined by the number of records that are possible in the queues. Information relating to the queues can be obtained by using the source collector QUEUE command. For information about the QUEUE command, see *Event Pump for z/OS Command Reference*.

For specific information about the data space startup parameters, see *Event Pump for z/OS Configuration Reference*.

4. If you plan to use multiple instances of Event Pump, each instance and its associated data space must include the following statement:

```
//ACC1IDxx DD DUMMY
```

where *xx* is a 2-character suffix that defines the instance. If you omit this DD statement, the ID used by Event Pump defaults to 01.

5. Copy the procedure to your PROCLIB data set.

For additional information about modifying the GTMDSPC procedure, see the procedure prolog.

Starting and stopping three address spaces

To monitor a z/OS operating system using three address spaces, the following Event Pump component address spaces must be running on the operating system to be monitored:

- Data space
- Event distributor
- Source collector

Run the components in the following order:

- Data space
- Event distributor
- Source collector

If you must stop the source collector or the event distributor, you must stop both of them, and then restart them. You do not have to stop and restart the data space.

If you must stop the data space, you must stop all three address spaces in the following order:

- Source collector
- Event distributor
- Data space

When you restart the data space, start the address spaces in the same start order:

- Data space
- Event distributor
- Source collector

When you stop the primary address space, any backup address space is also stopped.

Starting and stopping the data space

To start the data space as a started task (if the default sample procedure is used), enter the following command at the system console:

```
S GTMDSPC [,MBR = data_space_name]
```

In this example, *data_space_name* is the name of a member in the library pointed to by the ACC1PARM DD card in the data space started task procedure. The default name is PARMDSPC. This member contains the data space startup parameters.

The GTM5010I message is displayed within 1 minute after starting the primary data space address space and indicates that data space initialization is completed.

If you start a second copy of the data space to serve as a backup, the GTM4601I message indicates that the backup data space is monitoring the primary data space address space so that it can take over if a failure occurs.

Be sure to stop the source collector and the event distributor before you stop the data space. The data space does not shut down when the STOP (P) command is used unless both the source collector and the event distributor have stopped.

To stop the data space, enter the following command at the system console:

```
P data_space_name
```

Starting and stopping the event distributor

To start the event distributor as a started task (if the default sample procedure is used), enter the following command at the system console:

```
S GTMSRVR [,MBR=event_distributor_name]
```

In this example, *event_distributor_name* is the name of a member in the library pointed to by the AOPSYSIN DD card in the event distributor started task procedure. The default name is PARMSRVR. This member contains the event distributor startup parameters.

For V3 configuration users: The GTM4010I message indicates that event distributor initialization is completed. If you use the SNA protocol, the GTM7406I message indicates that the event distributor has connected to the Event Pump Windows servers. If you use TCP/IP, the GTM8205I and GTM8252I messages indicate that the event distributor has connected to the Event Pump Windows servers.

If you start a second copy of the event distributor to serve as a backup, the GTM4601I message indicates that the backup event distributor is monitoring the primary event distributor address space so that it can take over if a failure occurs.

You can stop and start the event distributor again without stopping and restarting the source collector and the data space.

To stop the event distributor, enter the following command at the system console:

```
P event_distributor_name
```

Starting and stopping the source collector

To start the source collector as a started task (if the default sample procedure is used), enter the following command at the system console:

```
S GTMPUMP [,MBR=object_pump_name]
```

In this example, *object_pump_name* is the name of a member in the library pointed to by the AOPSYSIN DD card in the source collector started task procedure. The default name is PARMPUMP. This member contains the source collector startup parameters.

For V3 configuration users: The GTM0990I message indicates that source collector initialization is completed. The GTM1770I and GTM1780I messages indicate that the source collector is receiving data from the Event Pump Windows servers. If you start a second copy of the source collector to serve as a backup, the source collector is monitoring the primary source collector address space so that it can take over if a failure occurs.

You can stop and start the source collector again without stopping and restarting the event distributor and the data space.

If the SHUTDOWN_CONFIRM startup parameter is set to YES, reply SHUT to the GTM9000I message.

To stop the source collector, enter the following command at the system console:

```
P object_pump_name
```

Appendix D. External Data Interface overview

The External Data Interface (EDI) enables external product applications (for example, AF/OPERATOR and System Automation for z/OS) to communicate with Event Pump. Using the EDI interface, external applications can communicate with Netcool/OMNIBus and TBSM servers that are running in the distributed environment. The EDI interface passes text-based messages between external applications and Event Pump. The EDI interface enables users to set format codes, action codes, and fields that are transformed and sent to the EIF probe by Event Pump.

The EDI message contains format, action, and field values. Each Event Pump event feed has a unique format code that defines the feed (for example Tivoli AF/OPERATOR uses Format=10 and Tivoli System Automation for z/OS uses Format=7). Each event feed has its own set of actions that can be passed to Event Pump. These actions reflect the state of the event feed. For example, AF/OPERATOR has the following values:

- **STATE_CHANGE** action value of 1
When the STATE_CHANGE action value is passed, the state of the AF/OPERATOR event feed is changed, for example the AF/OPERATOR address space is started, stopped, or abended.
- **MESSAGE** action value of 2
When the MESSAGE action is passed, the AF/OPERATOR data feed traps its message and passes it to Netcool/OMNIBus through Event Pump. In most cases these messages are informational messages.
- **EXCEPTION** action value of 3
The EXCEPTION action contains exception and resolution events. These are the critical (red) messages on Netcool/OMNIBus and require user attention.

Fields contain the data values for an event feed. Each field has a definition. For example, field 3 for AF/OPERATOR must contain an object name (started task or job), field 49 contains message text. Every data feed has a minimal set of required fields for a particular action. All other fields are added automatically by Event Pump. For example, AF/OPERATOR requires fields 3, 5, 8, 49 as mandatory fields for MESSAGE and EXCEPTION actions. All other fields are optional. For a list of all format, action, and field codes, review the GTMEDITB.REXX module.

There is an EDI REXX exit for external product REXX environments. Each event feed has an EDI exit:

EDI REXX exit Event feed

GTMEDISA	Tivoli System Automation for z/OS
GTMEDIOP	CA OPS/MVS
GTMEDIAF	Tivoli AF/OPERATOR
GTMEDIAO	BMC MainView AutoOPERATOR

You can include calls to these exits from the application address space REXX environment. See Figure 17 on page 174 for an example of the GTMEDIAF exit for the AF/OPERATOR event feed.

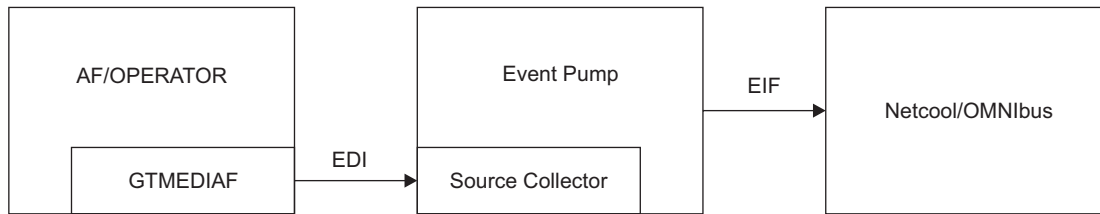


Figure 17. AF/OPERATOR EDI exit

EDI exit format

All EDI exits have similar format as shown in Figure 18.

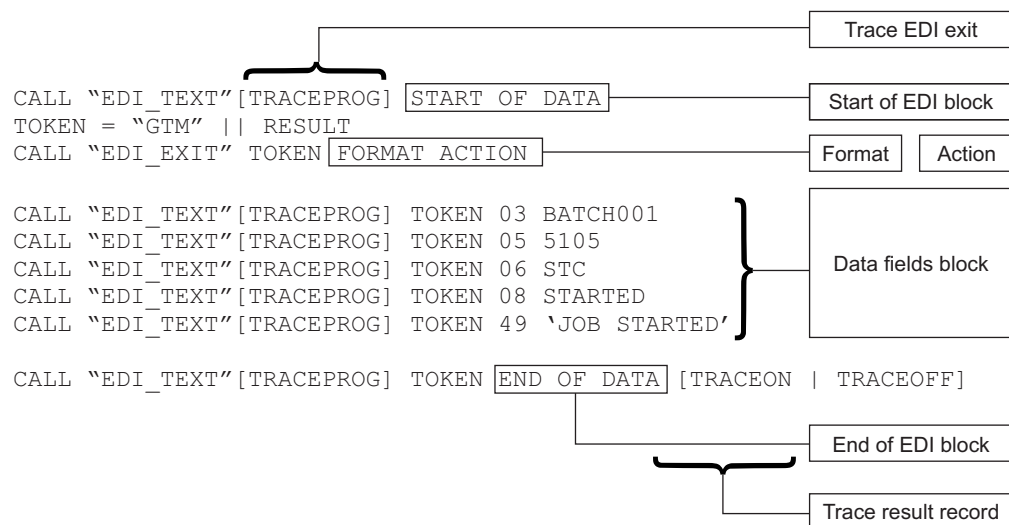


Figure 18. EDI exit CALL format

Each EDI API call must contain the following data blocks:

1. START_OF_DATA
2. Format type and action type
3. Data type and data values
4. END_OF_DATA

The first parameter that is passed to the EDI exit is the START_OF_DATA statement. This parameter initializes the EDI to start accepting information. TOKEN is the unique value that is used to distinguish between concurrent calls to the EDI API exit. With the use of the TOKEN value, data is not lost between calls to the EDI API exit. The token value must be prefixed with the GTM characters.

The second parameter contains the format type and action type for the EDI API exit. You can pass the format type and action type in numeric or character format.

The DATA FIELDS parameters specify the EDI API field number and field value. Format, action, and data type are passed in numeric or character format. You can use single quotation marks (') or double quotation marks (") when passing text data.

The END_OF_DATA statement is the last parameter that you pass to the EDI API exit. The END_OF_DATA data block notifies the EDI API exit that no more records follow for this request.

Each automation product has a slightly different method of invoking the EDI API exit. Table 55 lists the invocation methods by automation product.

Table 55. Methods of invoking the EDI API exit

Automation product	Method of invoking the EDI API exit
BMC MainView AutoOPERATOR for z/OS	CALL "GTMEDIA0" START_OF_DATA Token = "GTM" result CALL "GTMEDIA0" TOKEN 12 action_type
CA OPS/MVS	ADDRESS "TSO" "OI GTMEDIOP" START_OF_DATA Token = "GTM" rc "OI GTMEDIOP" TOKEN 09 action_type
Tivoli AF/OPERATOR	CALL "GTMEDIAF" START_OF_DATA Token = "GTM" result CALL "GTMEDIAF" TOKEN 10 action_type
Tivoli System Automation for z/OS	'GTMEDISA' START_OF_DATA Token = "GTM" rc 'GTMEDISA' TOKEN 07 action_type

Figure 19 is an example that posts the *Test message from SA390* message with a message ID of SAM001I to the BATCH001 resource:

```
'GTMEDISA' START_OF_DATA
TOKEN = "GTM" || result
'GTMEDISA' TOKEN 07 02          /* SA390 MESSAGE EVENT */
'GTMEDISA' TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TOKEN 03 BATCH001    /* OBJECT NAME */
'GTMEDISA' TOKEN 08 SAM001I     /* MESSAGE ID */
'GTMEDISA' TOKEN 49 "Test message from SA390" /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA
```

Figure 19. Example Tivoli System Automation for z/OS EDI API exit call

Sending notifications that your automation process is starting or ending is similar to the process of sending messages. The automation routine passes similar information to the REXX routine in the message process.

When automation begins, the EDI receives the following message:

Automation starting for object *resource_name*

When automation ends, the following message is sent to the EDI interface:

Automation ending for object *resource_name*

where *resource_name* specifies the resource name (for example DFHSM or HSC).

When the EDI API cannot send data to the source collector, the EDI API exit receives a return code greater than 0, and the GTMxx101 message is sent to Netcool/OMNIbus. When the EDI failure is resolved, a GTMxx102 message is sent to Netcool/OMNIbus.

Tracing CALL statements

The TRACEON keyword specifies that the EDI API exit prints the record that is sent to the source collector. This record is useful for troubleshooting a problem with the EDI API exit. The TRACEOFF statement disables this feature.

You can use the optional TRACEPROG keyword to trace the EDI API exit. You can trace all EDI API invocation statements or specific EDI API statements. Figure 20 shows an example of tracing all EDI API statements.

```
'GTMEDISA' TRACEPROG START_OF_DATA
TOKEN ="GTM" || result
'GTMEDISA' TRACEPROG TOKEN 07 02          /* SA390 MESSAGE EVENT */
'GTMEDISA' TRACEPROG TOKEN 05 IP01        /* SMF ID */
'GTMEDISA' TRACEPROG TOKEN 03 BATCH001    /* OBJECT NAME */
'GTMEDISA' TRACEPROG TOKEN 08 SAM001I     /* MESSAGE ID */
'GTMEDISA' TRACEPROG TOKEN 49 "Test message from SA390 " /* MESSAGE TEXT */
'GTMEDISA' TRACEPROG TOKEN END_OF_DATA
```

Figure 20. Tracing Tivoli System Automation for z/OS EDI API invocation statements

Figure 21 shows an example of tracing specific EDI API invocation statements.

```
'GTMEDISA' START_OF_DATA
TOKEN ="GTM" || result
'GTMEDISA' TRACEPROG TOKEN 07 02          /* SA390 MESSAGE EVENT */
'GTMEDISA' TOKEN 05 IP01                  /* SMF ID */
'GTMEDISA' TRACEPROG TOKEN 03 BATCH001    /* TRACE OBJECT NAME PROCESSING */
'GTMEDISA' TOKEN 08 SAM001I               /* MESSAGE ID */
'GTMEDISA' TOKEN 49 "Test message from SA390 " /* MESSAGE TEXT */
'GTMEDISA' TOKEN END_OF_DATA
```

Figure 21. Tracing Tivoli System Automation for z/OS specific EDI API invocation statements

API exit routine return codes

Table 56 shows the return codes from the EDI API exit routines.

Table 56. EDI API exit routine return codes

Return Code	Description
0	Normal Completion
4	EDI interface not available (source collector not running)
8	NOSPACE (Not enough space in the source collector EDI buffer for data)
12	Data item is larger than maximum allowed by the EDI interface (1024 bytes)
16	GETMAIN for working storage failed (PC routine)
20	GETMAIN for working storage failed (EDI routine)
24	An abend occurred, code in register 1 (EDI routine)
28	An abend occurred, code in register 1 (PC routine)
32	Zero length data passed

Format types, action types, and data types

Each data feed has a unique format type, action types, and data types:

- “BMC MainView AutoOPERATOR for z/OS (format type 12)”
- “CA OPS/MVS (format type 09)” on page 178
- “Tivoli AF/OPERATOR (format type 10)” on page 180
- “Tivoli System Automation for z/OS (format type 07)” on page 181

Note: In each of the following data feed descriptions, these are values that can be specified for the SEVERITY field. If no value is specified for the SEVERITY field, that value is set based on alerts.zos_severity_msgstate_lookup table or is set to MINOR if that table cannot be found.

Table 57. Severity specifications

Severity	Field value
Clear	CLEAR, UP, or 0
Indeterminate	IN, INDET, or 1
Warning	WA, WARNING, or 2
Minor	MI, MINOR, or 3
Major	MA, MAJOR, or 4
Critical	CR, CRITICAL, DO, DOWN, or 5

BMC MainView AutoOPERATOR for z/OS (format type 12)

BMC MainView AutoOPERATOR for z/OS uses a format type of 12 (AUTOOPERATOR) for the EDI exit. Table 58 is a list of action types and data types.

Table 58. BMC MainView AutoOPERATOR action types and data types

Action type	Field name	Data type	Value	Field type	Required	Comment
01 (STATE_CHANGE)	DATE_TIME	00	Date and time	CHAR	N	Format: yyyymmddhh:mm:ss.uuuuuu
	SEQ_NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data

Table 58. BMC MainView AutoOPERATOR action types and data types (continued)

Action type	Field name	Data type	Value	Field type	Required	Comment
02 (MESSAGE)	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data
03 ((EXCEPTION))	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data

CA OPS/MVS (format type 09)

Computer Associates CA OPS/MVS uses a format type of 9 (OPSMVS) for the EDI exit. Table 59 on page 179 is a list of action types and data types.

Table 59. CA OPS/MVS action types and data types

Action type	Field name	Data type	Value	Field type	Required	Comment
01 (STATE_CHANGE)	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data
02 (MESSAGE)	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data

Table 59. CA OPS/MVS action types and data types (continued)

Action type	Field name	Data type	Value	Field type	Required	Comment
03 (EXCEPTION)	DATE_TIME	00	Date and time	CHAR	N	Format: yyyymmddhh:mm:ss.uuuuuu
	SEQ_NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data

Tivoli AF/OPERATOR (format type 10)

Tivoli AF/OPERATOR uses a format type of 10 (AFOPERATOR) for the EDI exit.
Table 60 is a list of action types and data types.

Table 60. Tivoli AF/OPERATOR action types and data types

Action type	Field name	Data type	Value	Field type	Required	Comment
01 (STATE_CHANGE)	DATE_TIME	00	Date and time	CHAR	N	Format: yyyymmddhh:mm:ss.uuuuuu
	SEQ_NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data

Table 60. Tivoli AF/OPERATOR action types and data types (continued)

Action type	Field name	Data type	Value	Field type	Required	Comment
02 (MESSAGE)	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data
03 ((EXCEPTION))	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data

Tivoli System Automation for z/OS (format type 07)

Tivoli System Automation for z/OS uses a format type of 07 (SA390) for the EDI exit. Table 61 on page 182 is a list of action types and data types.

Table 61. Tivoli System Automation for z/OS action types and data types

Action type	Field name	Data type	Value	Field type	Required	Comment
01 (STATE_CHANGE)	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ_NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data
02 (MESSAGE)	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ_NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data

Table 61. Tivoli System Automation for z/OS action types and data types (continued)

Action type	Field name	Data type	Value	Field type	Required	Comment
03 ((EXCEPTION))	DATE_TIME	00	Date and time	CHAR	N	Format: <i>yyyymmddhh:mm:ss.uuuuuu</i>
	SEQ_NO	01	SMF-ID and AOMATNUM	INTEGER	N	SMF-ID concatenated with the trap match number associated with trap setting
	OBJECT_NAME	03	Object name	CHAR	Y	
	SMFID	05	SMFID of calling system	CHAR	Y	
	OBJECT_TYPE	06	Object type	CHAR	Y	
	SEVERITY	07	User-defined severity	CHAR or INTEGER	N	See Table 57 on page 177
	MESSAGE_ID	08	Message ID or exception name	CHAR	Y	
	MSG_TEXT	49				
	USER90 - USER139	90 - 139	User-defined fields	CHAR	N	Fields 90 -139 contain user-defined data

Appendix E. Event Pump IMS AO exits

To monitor IMS systems, the IHSIAOEX0 Type 2 exit that is supplied by Event Pump must be installed on those systems. This AO exit is required for the following reasons:

- The Event Pump must monitor many messages that the IMS subsystem issues only to the IMS master terminal (MTO). The IHSIAOE0 exit echoes some of these messages to the z/OS syslog, where they are trapped using NetView for z/OS message automation table services. The messages to be echoed are the messages present in the Event Pump IHSIAOEM message table, assembled as part of the AO exit installation.
- You can add, delete, or modify resources (programs, databases, logs, and subsystems) dynamically using the IMS /STA or /MODIFY commands. The IHSIAOE0 exit detects these commands and outputs messages to the z/OS syslog, where they are trapped using NetView for z/OS message automation table services.

The IHSIAOE0 exit is designed to only perform these two functions. No modifications or changes to any IMS commands or messages are performed by this exit. This exit can also exist with any user written IMS type 2 AO exit.

In many monitoring environments, the IHSIAOE0 exit alone is installed in the IMS subsystem. See Figure 22. This exit is link-edited with an alias of DFSAOE00, and IMS passes control to it as documented in the *IMS: Customization Guide*.



Figure 22. IHSIAOE0 Exit

However, if another product or you also require the use of the IMS AO exit interface, the Event Pump supplies an optional controller exit, named IHSIAOEC, that enables up to four other IMS AO type 2 exits to coexist with the IHSIAOE0 exit. See Figure 23. When the IHSIAOEC exit is given control by IMS, it passes control to each of the IMS AO exits in a way that exactly emulates how IMS passes control. No programming changes to the IMS AO exits are required.

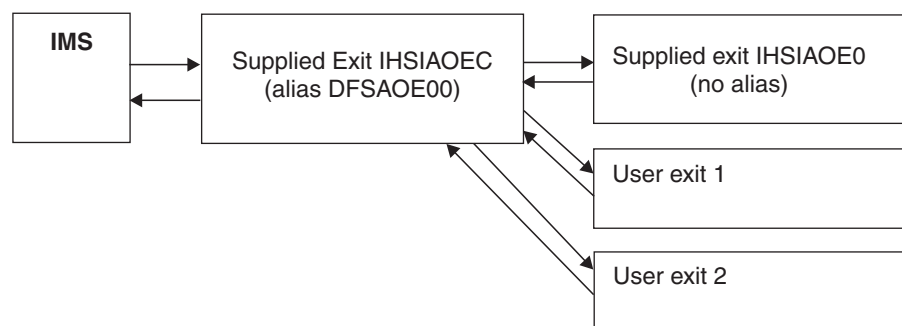


Figure 23. IMS AO Exit Interface

These exits are defined in the Event Pump IHSIAOEE exit control table, which is assembled during IHSIAOEC installation.

If no exits other than the Event Pump IHSIAOE0 exit requires the use of the IMS AO exit facility, the IHSIAOEC exit is not required for Event Pump operation.

IHSIAOE0

If the IHSIAOEC controller exit is not used, the IHSIAOE0 exit is link-edited with an alias of DFSAUE00.

When the IHSIAOE0 exit is loaded, either by the IMS program or by the IHSIAOEC controller exit, and it is initialized, the exit attempts to load the IHSIAOEM message table, which is assembled as part of the installation process. If this load fails, the IMS commands are still detected, but no IMS MTO messages are echoed to the z/OS syslog.

When the IHSIAOE0 exit is initialized, it checks all the messages that are generated and the commands that are issued. It uses only the first segment (approximately 70 bytes) of an IMS MTO message or command. Subsequent segments are presented by the IMS program or the AO IHSIAOEC controller exit.

The following items determine the reply value that is returned to the IMS program by the IHSIAOE0 exit:

- To enable coexistence, if an AO type 1 DFSAUE00 exit is present, the IHSIAOE0 exit sets the reply value to indicate that this first segment and any subsequent segments are to be presented by the IMS program to the type 1 exit. The IMS program does not present any subsequent segments for this MTO message or command to the IHSIAOE0 exit.
- If no AO type 1 DFSAUE00 exit is present, the IHSIAOE0 exit sets the reply value to indicate that any subsequent segments are not to be presented.

IHSIAOEC controller exit

If the IHSIAOEC controller exit is used, it is link-edited with an alias of DFSAUE00, and the IHSIAOE0 exit is not link-edited with this alias.

When the IHSIAOEC exit is loaded by the IMS program and initialized, it attempts to load the IHSIAOEE exit control table, which is assembled as part of the installation process. If the load fails, the IHSIAOEC exit does not act when the IMS program presents MTO message and command segments.

The loaded exit control table holds a list of AO type 2 exits that are to be loaded and controlled by the IHSIAOEC exit (in order). Each exit that is specified in the table is loaded and, providing the load was successful, the entry address is saved, and storage areas are obtained. The IHSIAOEC exit then provides each controlled exit an opportunity for initialization (emulating IMS processing, which passes control to the IMS AO type 2 exit for initialization), and returns to the IMS program.

When initialized, the IHSIAOEC exit calls each AO type 2 exit that it has loaded (in order), and emulates IMS processing for each exit. For example, all AO type 2 exits are able to process as if the IHSIAOEC exit were not calling them. For each MTO message or command segment that is presented by the IMS program, the IHSIAOEC exit mirrors the IMS program in providing each exit an opportunity to

perform its function and then returns to the IMS program. The IHSIAOEC exit does not use or change the IMS MTO message or command segment data.

Only one client exit can set the AOE0RPLY value to be sent back to the IMS program. You determine which exit by setting a flag in the IHSIAOEE table for that exit. This flag determines the AOE0RPLY value. If no exit is supplied to set the AOE0RPLY value, the IHSIAOEC exit sets it in a similar way to the IHSIAOE0 exit.

Although they are not required for Event Pump operation, you can use both IMS AO exits supplied with the Event Pump with IMS FDBR regions.

Appendix F. Migrating from TBSM Version 3 object pump to Event Pump Version 4 Release 2

Event Pump consists of the source collector and the data space event distributor address spaces on each z/OS image. The basic pump operation remains unchanged. The following updates were made to support a stand-alone operation of the Event Pump for use with the Tivoli Business Service Manager (TBSM) Version 4 Release 2 product:

- Event Pump initialization now occurs without requiring a connection to the TBSM Version 3 Event Handler.
- The source/390 object pump also used two-way communication with the TBSM Version 3 server for message registration including standard console message traps and generic console message traps. The registration data is now obtained from the GTMINPUT DD file and is allocated to the source collector address space on each z/OS image.
- TBSM Version 3 server registration data can be captured to a file that is pointed to by the GTMOUT DD and converted to the new format by using an offline utility to facilitate migration to stand-alone mode.
- The new facilities are dormant until they are enabled by the inclusion of specific DD statements into the source collector startup JCL.
- The stand-alone registration data is in a readable format so that it can be edited and modified.
- You can use the REGISTER modify command to selectively register and unregister monitored resources by type or by user-defined groups without restarting the source collector address space. *Event Pump for z/OS Command Reference* contains additional information about the REGISTER command.

Summary of changes for IBM Tivoli Event Pump for z/OS Version 4.2.1

These new or changed functions were available with IBM Tivoli Event Pump for z/OS Version 4.2.1:

- Cross-sysplex registration and clearing of events.
- Consolidation into one package of previously delivered event feed support, including CICS Transition Data Queue, Tivoli Workload Scheduler (TWS), Resource Manager Facility (RMF III), and CICSplex System Manager (CICSplex SM).
- Improved throughput performance of event processing by using the customizer.
- The NetView for z/OS product is no longer required for processing the DB2 event feed.
- Enhanced capabilities in the customizer so that you can process incoming data source messages without additional coding to format the OMNIbus event.

For a summary of changes for IBM Tivoli Event Pump for z/OS Version 4.2.2, see Chapter 2, “New or changed functions for IBM Tivoli Event Pump for z/OS Version 4.2.2,” on page 9.

Sysplex support

Sysplex support within TBSM Version 3.1 was provided within the TBSM server running on the Windows operating system components that distributed trap registration data to all Event Pumps within a sysplex. The removal of the TBSM server running on the Windows operating system means that this capability is no longer available within the new stand-alone Event Pump environment. Function was added to Event Pump relating to dynamically created message and event traps.

Registration data migration

A registration data capture and conversion utility is provided to facilitate the migration of Event Pump to a stand-alone operation.

You can capture registration data received from the distributed components on each host as the data is received by Event Pump on that host system. The captured raw registration data can then be processed by an offline utility to convert the data to a readable format that is required by the stand-alone facility. Also, see “Migration scenario: migrating registration data from TBSM Version 3 to Event Pump V4.2” on page 191.

Enabling registration data capture

To capture the registration data, create a sequential data set with the following attributes:

```
RECFM=VB,LRECL=2000,BLKSIZE=2004
```

You might have to experiment with the space allocation, but several cylinders is sufficient to capture the bulk of the registration data. When the data set is full, Event Pump stops collecting registration data but continues operation.

Add the GTMOUT DD statement to the Event Pump startup JCL and point it to the preceding data set. You must allocate this data set with DISP=MOD specified. Ensure that the user ID under which the pump address space runs has update authority to the data set allocated to the GTMOUT DD file.

You might want to add a step to the Event Pump startup JCL to empty the data set before Event Pump operation. Because the data set is allocated with DISP=MOD, you must shut down Event Pump before you can read the data. Then, you can start the Event Pump components. Event Pump writes all registration data it receives to the GTMOUT file.

Converting the captured registration data

The Registration Data Conversion Utility sends the data captured by Event Pump to the GTMOUT DD file and converts the data into the registration format that is required by Event Pump when running in stand-alone mode.

The conversion operation is controlled by a set of IBM supplied control data in the GTMCNVCT member of the SGTMSAMP sample. The GTMCNVRD member in the SGTMSAMP file in the TBSM installation library contains sample JCL to run the utility.

Create a sequential data set (LRECL=80, BLKSIZE=*size* RECFM=FB) to hold the output from the conversion utility.

Modify the GTMCNVRD JCL to allocate the output data set that you created to the GTMOUT DD file, and the registration data capture file from the Event Pump operation to the GTMIN DD statement. Then, run the job to process the registration data.

Adding group entries

Optionally, you can modify the generated registration data to group-specific entries together. You can add GROUP statements to the resultant file to make manual registration operations easier when using the REGISTER console command, or to limit the entries that are automatically registered during Event Pump startup by using the REGISTER startup parameter.

Activating the new registration data

Remove the GTMOUT DD statement from the Event Pump startup JCL. You can allocate it to the SYSOUT DD file so that you can compare the registration data created when running in stand-alone mode with the data originally captured from the distributed system. These data can be used as a validation mechanism for the conversion process.

Migration scenario: migrating registration data from TBSM Version 3 to Event Pump V4.2

Complete this task to forward data from TBSM Version 3 to the source collector for Event Pump. Before you begin this task, ensure that TBSM Version 3 is running on the host system.

Use the following steps to import TBSM Version 3 data to the Event Pump source collector:

1. Collect the data and save it in a data set.
2. Add a DD statement similar to this statement to the source collector procedure:

```
//GTMOUT DD DSN=TDTBSM.TBSM31.GTMOUT,DISP=(MOD,CATLG,CATLG),
//          DCB=(LRECL=2000,BLKSIZE=2004,RECFM=VB),
//          UNIT=SYSDA,VOL=SER=DPQTOA,SPACE=(CYL,(1,1),RLSE)
```

In this example, the data set name is TDTBSM.TBSM31.GTMOUT. You specify the data set name and size according to your requirements. This data set contains raw registration data.

3. Shut down TBSM Version 3.
4. Convert the data set to a file for the source collector with the GTMCNVRD batch job as shown in the following example:

```
//GTMCNVRD JOB (MYJOB,B510,MYJOB),'*MYID*',
//          MSGCLASS=X,CLASS=A,MSGLEVEL=(1,1)
//*
//IRXJCL EXEC PGM=IRXJCL,PARM='GTMRX044',REGION=0K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//SYSEXEC DD DISP=SHR,DSN=TDTBSM.V4R1M0.SGTMEEXEC
//GTMIN DD DISP=SHR,DSN=TDTBSM.TBSM31.GTMOUT
//GTMOUT DD DISP=SHR,DSN=TDTBSM.CONVERT.LIB(NMP113)
//GTMCNTRL DD DISP=SHR,DSN=TDTBSM.TBSM31.JCL(GTMCNVCT)
//*
```

Usage notes:

- a. The GTMIN DD statement defines the data set that you convert to a file for the source collector.

- b. The GTMOUT DD statement defines the file that you create from the data set. In this case, an existing page data set (PDS) with 80-byte records.
5. Insert the GTMINPUT DD statement into the source collector job. For example:
- ```
//GTMINPUT DD DISP=SHR,DSN=TDTBSM.CONVERT.LIB(NMP113)
```

TDTBSM.CONVERT.LIB(NMP113) contains converted registration readable by the pump.

In the example, the TDTBSM.CONVERT.LIB(NMP113) file is supplied to the source collector.

---

## Startup considerations for TBSM Version 3 and Event Pump Version 4.2 (dual mode)

These are parameters and system variables for Event Pump. These parameters can be used with dual mode, so that you can send events to the Version 3.1 server and to the EIF probe.

See the following sections for the source collector, data space, and event distributor startup parameters:

- “Event Pump source collector startup parameters for TBSM Version 3 and Event Pump Version 4”
- “Event Pump data space startup parameters for TBSM Version 3 and Event Pump Version 4” on page 193
- “Event Pump event distributor startup parameters for TBSM Version 3 and Event Pump Version 4” on page 194

In some cases, you might have to modify the startup parameters that are provided with Event Pump. To modify the startup parameters, copy the members that you have to modify from the SGTMSAMP data set to a user-defined data set and then modify the copied members. This process preserves the original member and protects your modifications when maintenance is applied.

### Event Pump source collector startup parameters for TBSM Version 3 and Event Pump Version 4

The parameters in this section can be used with dual mode, where you can send events to the Version 3.1 server and to the EIF probe.

The Event Pump source collector startup parameters are supplied through the AOPSYSIN DD statement (LRECL=80, RECFM=F or FB). Most startup parameters are optional and have default settings if the settings are not specified by the user. If startup parameters that are required are omitted, some functions of the source collector might not be available.

Do the following steps:

1. Copy the GTMPUMP started task from the SGTMSAMP library to a user-defined system PROCLIB.
2. Copy the PARMPUMP startup parameter member from the SGTMSAMP library to a user-defined data set that is referenced by the source collector DD statement in the startup job control language (JCL).
3. Update the PARMPUMP startup parameter member with the following parameters:

```
*
** Comments start with an asterisk in Column One
```

```

SUBSYSTEM_NAME=TM.. | name
CONSOLE=TM390.. | name
CONSOLE_MSCOPE=LOCAL | ALL
MAX_TRAPS=2016 | number
TERMINAL_PREFIX=TM390 | prefix
NUMBER_OF_TERMINALS=0 | n
MODEL2_LOGMODE=D4A32782 | name
MODEL3_LOGMODE=D4A32783 | name
MODEL4_LOGMODE=D4A32784 | name
MODEL5_LOGMODE=D4A32785 | name
LOG_SCREEN=NO | YES
EDI_BUFFER_SIZE=2048 | size
EXEC_REFRESH=YES | NO
SUBSYSTEM_MAXTHREADS=2 | threads
MAXTHREADS_PROMPT=YES | NO
TRAP_REPORT=SYSOUT(A) | sysout(class) | dsname
LOG_FILE=SYSOUT(A) | sysout(class) | dsname
LOG_FILE_RECORD_LENGTH=200 | length
LOG_SPIN_INTERVAL=04:00:00 | hh:mm:ss
TRACE=YES | NO
COMMAND_PREFIX=subsystem_name | prefix
SA390_JOBNAME=jobname
OPSMVS_JOBNAME=jobname
SHUT_OBJECT_SERVER=YES | NO
SHUTDOWN_DELAY=n | 0
SHUTDOWN_CONFIRM=YES | NO
RMF_SERVER_IP=ip address
WTOR_ALERTS=NO | YES
WTOR_DELAY=30 | nnn
WTOR_CLEANUP=900 | nnn
PPI=NO | YES
PPI_BUFFER_SIZE=512 | nnn
PPI_BUFFER_QUEUE_LIMIT=1000 | nnn
TCPIP_TASKNAME=taskname
PPI_TRACE=NO | YES
SBCS_CODEPAGE=0037|nnnn
DBCS_CODEPAGE=nnnn
LANGUAGE_ID=ENU|xxx
SYSPLEX=name
GTFTRACE=YES |NO
REGISTRATION_WAIT_TIME=00:00:00 | hh:mm:ss
REGISTRATION_TIMEOUT=INFORM | RETRY
REGISTRATION_RETRY_LIMIT=3 | 1-10

```

For information about the startup parameters, see *Event Pump for z/OS Configuration Reference*.

## Event Pump data space startup parameters for TBSM Version 3 and Event Pump Version 4

Do the following steps:

1. Copy the GTMDSPC started task from the SGTMSAMP library to a user-defined system PROCLIB.
2. Copy the PARMDSPC startup parameter member from the SGTMSAMP library to a user-defined data set that is referenced by the data space DD statement in the startup job control language (JCL).
3. Update the PARMDSPC startup parameter member with the following parameters:
 

```

OSQUEUE SIZE=osqueue size
OPQUEUE SIZE=opqueue size
LOGQUEUE SIZE=logqueue size

```

For information about the startup parameters, see *Event Pump for z/OS Configuration Reference*.

## Event Pump event distributor startup parameters for TBSM Version 3 and Event Pump Version 4

Do the following steps:

1. Copy the GTMSRVR started task from the SGTMSAMP library to a user-defined system PROCLIB.  
There are two members in the SGTMSAMP library that can be used with the GTMSRVR procedure:
  - PARMSRIP member that contains a sample set of IP control statements
  - PARMSRVR member that contains a sample set of LU6.2 control statements
2. Copy the PARMSRIP and the PARMSRVR startup parameter members from the SGTMSAMP library to a user-defined data set that is referenced by the event distributor DD statement in the startup job control language (JCL).
3. Update the PARMSRIP and the PARMSRVR startup parameter members with the following parameters:

\*Comments start with an asterisk (\*) in column one.

```
PROTOCOL=LU62 | IP
REMOTE_APPLID=remote_applid
LOCAL_LUNAME=local_luname
LOGMODE=LU62PS | logmode
TCPIP_ADDRESS=tcip_address
TCPIP_NAME=tcip_name
TCPIP_PORT=port_number
TCPIP_SERVER_ADDRESS=tcip_server_address
TCPIP_SERVER_NAME=tcip_server_name
TCPIP_SERVER_PORT=port_server_number
TCPIP_JOBNAME=tcip_jobname
TIMEOUT=30 | timeout_value
CODEPAGE=037 | codepage
BUFFERSIZE=1 | buffersize
TCPIP_ADDRESS_BACKUP=tcip_backup_address
TCPIP_NAME_BACKUP=tcip_backup_name
TCPIP_PORT_BACKUP=tcip_backup_port
RETRY_INTERVAL=30 | retry_interval
RETRY_COUNT=0 | retry_count
VALIDCLIENT=client_ip_address | client_hostname
VALIDCLIENT_HOSTRES=YES | NO
LOG1=log_dataset1
LOG2=log_dataset2
TRACE=YES | NO
REGTRACE=YES | NO
REQUEST_RESPMSG=YES | NO
EIF_CONF=/'%h1q.SGTMSAMP(PARMEIF)'
EIF_LOG=Y | N
EIF_LOG_LEVEL=TBSM | TEC | BOTH
```

For information about the startup parameters, see *Event Pump for z/OS Configuration Reference*.

The following parameters are used for LU6.2 and TCP/IP interfaces for TBSM Version 3 and Event Pump Version 4:

- EIF\_CONF
- EIF\_LOG
- EIF\_LOG\_LEVEL
- LOG1

- LOG2
- RETRACE
- TRACE

The following parameters are used for LU6.2 for TBSM Version 3 and Event Pump Version 4:

- LOCAL\_LUNAME
- LOGMODE
- REMOTE\_APPLID

The following parameters are used for TCP/IP interface for TBSM Version 3 and Event Pump Version 4:

- BUFFERSIZE (optional)
- CODEPAGE (optional)
- REQUEST\_RESPMSG (optional)
- resolvertimeout/resolverudpretries (optional for recovery)
- RETRY\_COUNT (optional for recovery)
- RETRY\_INTERVAL (optional for recovery)
- TCPIP\_ADDRESS
- TCPIP\_ADDRESS\_BACKUP (optional for recovery)
- TCPIP\_JOBNAME (optional)
- TCPIP\_NAME\_BACKUP (optional)
- TCPIP\_PORT
- TCPIP\_PORT\_BACKUP (optional for recovery)
- TCPIP\_SERVER\_ADDRESS (optional for recovery)
- TCPIP\_SERVER\_NAME (optional for recovery)
- TCPIP\_SERVER\_PORT
- TIMEOUT (optional)
- traceresolver (optional for recovery)
- VALIDCLIENT
- VALIDCLIENT\_HOSTRES

If the IP address of the Listener is set up in the Network Address Table (NAT) configuration, the NAT IP address must be used. If the NAT IP address is not used, the event distributor does not connect to the EIF probe.

## Optional DD names for the TBSM event distributor for TBSM Version 3 and Event Pump Version 4

You can use optional DD names in the event distributor started task (GTMSRV) when you run the TCP/IP interface.

The following JCL control statement is used when the TCP/IP is not configured properly or when you are experiencing host resolution problems:

```
//SYSTCPD DD DSN=tcip_dataset(TCPDATA),DISP=SHR
```

The following TRACE and LOG DD names provide you with additional diagnostic data that shows the TBSM messages that were transmitted over an IP address:

```
//TRACE DD SYSOUT=*
```

```
//LOG DD SYSOUT=*
```

## Setting parameters for language support for TBSM Version 3 and Event Pump Version 4

Set the following parameters for language support:

- For Windows systems:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Accessible Software, Inc. \Access1\1.0\Settings]
```

- DefaultLanguage=*language\_id*

The valid language IDs are the directory names available in  
tivoli\manager\lang

- LocalCodePage=*NT\_code\_page\_id*

For example, 1252 for Danish.

- RemoteCodePage=*mainframe\_code\_page*

For example, 20277 for Danish.

If you are using a code page other than 437 on your servers, review the comments in the cconv.tbl file in the tivoli\manager\lang\ *language\_id* directory. The console code page is set by the SetConsoleCodePage command.

Issue the SetConsoleCodePage command and note the value of Current Console Code Page:

- For z/OS systems:

```
LANGUAGE_ID = SBCS_CODEPAGE=
```

---

## Filtering Event Pump for z/OS events for TBSM Version 3 and Event Pump Version 4

You can perform further filtering of z/OS events to the EIF probe from the Event Pump program. Event messages of interest are intercepted within the event distributor as the messages are transmitted to the EIF probe by using the Event Integration Facility (EIF).

### Event format

All events built by the interface contain the following standard slots:

#### **name\_value\_pairs**

This slot contains all the data from the Event Pump event record in a format that enables the Event Pump EIF event listener to change the event into the format that is expected by Event Pump.

**source** This slot contains the information from the host systems VTAM® **netname.cpname** field. This information enables the Event Pump EIF event listener to place the changed Event Pump event onto the correct operating system queue within Event Pump.

Data from the fields within the Event Pump event record can be included into slots that you defined within the event. Various formats are available to enable the selection and extraction of data from the Event Pump event record field.

### Message selection and filtering within the event distributor

To set up message selection and filtering, create an EIF select table in the GTMEIFSL data set that is read by the event distributor. You can refresh the table dynamically without recycling the event distributor. The EIF select table is used to select records that are based on the format type, action type, and data within the fields of each Event Pump event record. Each select table rule specifies where the record is sent: Netcool/OMNIBus, Event Pump, both, or neither.

Use the following statements to code your EIF select table:

- DEFAULT
- GROUP

- ONSELECT
- SELECT

For information about the EIF select table statements, see *Event Pump for z/OS Configuration Reference*.

## Event logging

Use the EIF event logging facilities to help construct your EIF select table and to debug event problems.

Start and stop event logging by specifying the EIF\_LOG event distributor and EIF\_LOG\_LEVEL startup parameters, or by using the event distributor EIF LOG modify command. If you enable EIF event logging within the event distributor, the events are not displayed on the operator console. They are written to the AOPLOG data set.

As each SELECT statement is processed, it is internally assigned a sequential number, starting with 1. When logging is enabled, this number is used to identify the SELECT statement that matched an Event Pump event record.

Enable event logging to display the following events:

- The generated event, and return code from the call to the EIF sender
- The Event Pump event record, the default or SELECT entry number that selected it, and the Event Pump and the Netcool/OMNIbus options in effect for the entry that selected the event record
- Both of the preceding events

The level of detail that is logged is controlled by the event distributor EIF\_LOG\_LEVEL startup parameter or the event distributor EIF LOG LEVEL modify command.

Use logging to view the following results:

- View the fields and data within each Event Pump event record.
- View the result of the EIF select table matching criteria. Use this result to determine the SELECT statement that matched the event record and also the Event Pump and the Netcool/OMNIbus options that are in effect.
- View the generated Netcool/OMNIbus event and the return code from the call to EIF that sent the event to Netcool/OMNIbus.

In the example in Figure 24 on page 198, the first SELECT statement matched the data in the Event Pump event record. This event is sent only to Netcool/OMNIbus.

```

EIF0201 : TBSM EVENT RECORD FOLLOWS
FORMAT=99 ACTION=98
03=TEST4
46=IEF403I
49=JOB CICSPROD STARTED AT 12:45:59
EIF0202 : END OF TBSM EVENT RECORD
EIF0204 : SELECTED BY SELECT 001 TEC=YES TBSM=NO
EIF0301 : TEC EVENT DATA FOLLOWS
TEC_Generic;source=P390.P390SSCP;name_value_pairs=
'_FORMAT=99','_ACTION=98','_F03=TEST4','_F46=IEF4
_03I','_F49=JOB CICSPROD STARTED AT 12:45:59';IDEN
T='TEST4';END
EIF0302 : END OF TEC EVENT DATA
EIF0303 : EIF RC=00000000

```

Figure 24. Sample EIF Log Output (EIF\_LOG\_LEVEL=BOTH)

## Implementation and configuration on the z/OS host

To configure the Event Integration Facility (EIF) within the Event Pump event distributor, you must do the following steps:

- Update the Event Pump event distributor JCL.
- Update the Event Pump event distributor security profile.
- Update the EIF configuration file.
- Update the Event Pump event distributor startup parameters.

### Updating the Event Pump event distributor JCL

Add the following DD statements to each Event Pump event distributor startup JCL member that uses the interface:

- //GTMEIFSL: Points to a data set that contains the SELECT statements used to select the records to send to the EIF probe
- //GTMEIFSP: SYSOUT. Used to report any errors with the SELECT statements
- //SYSPRINT: SYSOUT. Messages from the EIF interface
- //SYSOUT: SYSOUT. Messages from the EIF interface

You might also have to include the SYS1.SCEERUN library in the STEPLIB DD statement if this library is not available through your link list concatenation. In this case, the library must be authorized program facility (APF) authorized, otherwise all libraries in the STEPLIB concatenation are unauthorized, and the Event Pump event distributor cannot initialize.

### Updating the EIF configuration file

Do the following steps to update the EIF configuration file:

1. Copy the PARMEIF member from the SGTMSAMP library to a user-defined data set with a low-level qualifier of SGTMSAMP.
2. Modify the EIF\_CONF statement in the PARMSRIP or PARMSRVR members (copied from the SGTMSAMP library to a user-defined library) to reference the PARMEIF member. The EIF\_CONF statement can be in the HFS or the PDS.
3. Modify the statements in the PARMEIF member as required.

The ServerLocation and ServerPortv3 statements are required; the other statements are optional and have default settings if not specified. For information about the EIF configuration file statements, see *Event Pump for z/OS Configuration Reference*.

## Updating the Event Pump event distributor startup parameters

To specify the EIF configuration file path and name to the Event Pump event distributor, add this statement to the startup parameters:

EIF\_CONF=/path/filename

To specify the event logging option to the Event Pump event distributor, add this statement to the event distributor input parameters:

EIF\_LOG=Y | N

This option turns logging of Event Pump and Netcool/OMNIbus events on or off. The EIF\_LOG\_LEVEL parameter controls the level of detail that is logged. Logged data is written to the AOPLOG DD file. If this option is not specified, the default setting is EIF\_LOG=N. Logging can also be controlled by using the following event distributor EIF LOG LEVEL modify command:

EIF\_LOG\_LEVEL=TBSM | TEC | BOTH

This parameter can be used to control the level of detail that is logged to the AOPLOG DD file. For information about the startup parameter statements, see *Event Pump for z/OS Configuration Reference*.

## The event distributor operator commands for the EIF interface for TBSM Version 3 and Event Pump Version 4

The event distributor supports the following modify commands for the EIF interface:

### EIF STOP

Stops the sending of events to Netcool/OMNIbus. Events are not cached or stored while the interface is stopped. Missed events are lost. All events are sent to Event Pump.

### EIF START

Restarts the EIF interface after a STOP. The event distributor sends selected records to Netcool/OMNIbus.

### EIF REFRESH

Causes the event distributor to read the SELECT statement input data set that is allocated to the GTMEIFSL DD statement and to build a new SELECT table. This command can be used to change the select table without having to recycle the event distributor.

### EIF STATUS

Displays information about the status of the EIF interface, the connection to Netcool/OMNIbus, and statistics for the records that are sent to Netcool/OMNIbus and the Event Pump.

This value is the default value.

### EIF\_LOG YES | NO | STATUS

Turns logging of Event Pump and Netcool/OMNIbus events to the AOPLOG file ON or OFF or displays the current setting for the EIF LOG option.

The default is STATUS.

### EIF\_LOG\_LEVEL TBSM | TEC | BOTH

Controls the level of logging of Event Pump and Netcool/OMNIbus events passing through the interface. If the option is missing or not valid, it is ignored and the current log level is displayed.

## Supported data sources and event feeds

Table 62 provides comparison information for the supported data sources for TBSM Version 3 and Event Pump Version 4. Table 63 provides comparison information for the supported event feeds for TBSM Version 3 and Event Pump Version 4

*Table 62. Supported data sources*

| Data sources                                | TBSM 3.1 | Event Pump 4.2.2 |
|---------------------------------------------|----------|------------------|
| IBM                                         |          |                  |
| JOBLOG files                                | No       | Yes              |
| SYSLOG messages (including WTOR processing) | Yes      | Yes              |

*Table 63. Supported event feeds*

| Event Feeds                                        | TBSM 3.1 | Event Pump 4.2.2 |
|----------------------------------------------------|----------|------------------|
| IBM                                                |          |                  |
| CICS Base                                          | Yes      | Yes              |
| CICSplex System Manager                            | Yes      | Yes              |
| CICSplex Transient Data Queue                      | Yes      | Yes              |
| Data Facility Storage Management Subsystem (DFSMS) | Yes      | Yes              |
| Data Facility Hierarchical Storage Manager (DFHSM) | Yes      | No               |
| DB2                                                | Yes      | Yes              |
| Health Checker for z/OS                            | No       | Yes              |
| IMS                                                | Yes      | Yes              |
| Resource Management Facility (RMF)                 | Yes      | Yes              |
| Resource Object Data Manager (RODM)                | Yes      | No               |
| System Automation (SA) for z/OS                    | Yes      | Yes              |
| Tivoli AF/Operator on z/OS                         | Yes      | Yes              |
| Tivoli OMEGAMON Classic for CICS                   | Yes      | No               |
| Tivoli OMEGAMON Classic for DB2                    | Yes      | No               |
| Tivoli OMEGAMON Classic for IMS                    | Yes      | No               |
| Tivoli OMEGAMON Classic for z/OS                   | Yes      | No               |
| Tivoli Workload Scheduler (TWS)                    | Yes      | Yes              |
| WebSphere for z/OS                                 | Yes      | No               |
| ASG-TMON                                           |          |                  |
| ASG-TMON for CICS TS                               | Yes      | No               |
| ASG-TMON for DB2                                   | Yes      | No               |
| ASG-TMON for z/OS                                  | Yes      | No               |
| ASG-Zeke                                           | Yes      | No               |
| BMC                                                |          |                  |
| BMC MainView AutoOPERATOR for z/OS                 | Yes      | Yes              |
| BMC MainView Control-M for z/OS                    | Yes      | No               |
| BMC MainView for CICS                              | Yes      | Yes              |

Table 63. Supported event feeds (continued)

| Event Feeds                              | TBSM 3.1 | Event Pump 4.2.2 |
|------------------------------------------|----------|------------------|
| BMC MainView for DB2                     | Yes      | Yes              |
| BMC MainView for IMS                     | Yes      | No               |
| BMC MainView for z/OS                    | Yes      | Yes              |
| Computer Associates (CA)                 |          |                  |
| Computer Associates: CA 7 Job Management | Yes      | No               |
| Computer Associates: CA JobTrac          | Yes      | No               |
| Computer Associates: CA OPS / MVS        | Yes      | Yes              |



---

## 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 web sites are provided for convenience only and do not in any manner serve as an endorsement of those web

sites. The materials at those web sites are not part of the materials for this IBM product and use of those web sites 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 wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
2Z4A/101  
11400 Burnet Road  
Austin, TX 78758  
U.S.A.

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.

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.

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

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

---

## Trademarks

IBM, the IBM logo, and [ibm.com](http://www.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 <http://www.ibm.com/legal/copytrade.shtml>.

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows 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.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, and service names may be trademarks or service marks of others.



---

# Index

## Special characters

\$\$INSYS0 76

## A

- accessibility xiv
- adding group entries 191
- address spaces
  - data space 7
  - descriptions 4
  - dispatching priority 20
  - master 4
  - z/OS performance 20
- AF/OPERATOR EDI Interface
  - installation 53
- allocating
  - log data sets 21
- AO exit 106
- AOPCICS2 160
- AOPCICS3 160
- AOPSYSIN DD 33, 165, 166
- APF authorization 19
  - dynamic changes 20
- automation
  - SMF record type 30 163
  - starting and ending 54, 61, 66, 71
- automation event feed 53
- automation external data interface 53
- AutoOPERATOR interface
  - installation 65

## B

- batch jobs, detection 11
- BMC MainView AutoOPERATOR 65
- BMC MainView for CICS 77
  - message mask 149
  - monitor file 34
- BMC MainView for DB2 80
  - message mask 149
  - monitor file 34
- BMC MainView for z/OS 75
  - message mask 149
  - monitor file 34
- BMC MainView performance monitors
  - BMC MainView for CICS 77
  - BMC MainView for DB2 80
  - BMC MainView for z/OS 75
- books
  - See publications

## C

- CA OPS/MVS 107
- CICS subsystem data resources 83
- CICSplex SM Web User Interface 87
- CICSplex SP file 159
- CICSplex System Manager 32, 85, 164
  - message mask 149

CICSplex System Manager (*continued*)

- monitor file 34
- CNMPNL1 124
- CNMPNL2 124
- CNMPROC 58
- commands
  - command line interface 152
  - SQL 151
- components
  - data space address space 7
  - event distributor 5
  - event processing 6
  - external data interface (EDI) 6
  - master address space 4
  - SOAP interface 6
  - source collector 5
- Computer Associates OPS/MVS 69
- configuring
  - EIF probe 29
  - Netcool/OMNIBus ObjectServer 26
  - three address spaces 163
  - two address spaces 31
- conventions
  - environment variables xvi
  - path names xvi
  - syntax xvi
  - typeface xv
- CPEDEFLT 34, 159
- CPESTRES 34, 159
- CPSM 164
- customizer 133
- customizing event feeds for NetView for z/OS 128

## D

- data
  - activating new registration data 191
  - converting captured registration data 190
  - enabling registration data capture 190
  - registration data migration 190
  - registration data migration scenario 191
- data sources 45
  - overview 7
  - SYSLOG 50
- data space
  - starting 39, 169
  - startup parameters 36, 168, 193
  - stopping 39, 169
- data space address space 7
- data space startup procedure 36, 168
- DB2 91
  - installing 91
  - message mask 149
  - monitor file 34
  - security 12, 92
- DB2 monitoring 91
  - DB2 event processing 93
    - disabling 93
    - enabling 93
  - DB2 sysplex support 91
  - excluding resources and events 94

- DB2 monitoring (*continued*)
  - hierarchy of exclude processing 95
  - process 95
  - reasons for 94
  - refreshing exclude rules 96
- DB2 resources
  - excluding resources and events 94
  - hierarchy of exclude processing 95
  - process 95
  - reasons for 94
  - refreshing exclude rules 96
- DB2 Universal Database for z/OS 91
- DD concatenations 124
- DD names, optional 195
- DEFAULT statement 137
- DEFLOGS 159
- DEFLOGS job 21
- detecting resources 11
- detection process
  - batch jobs 11
  - by event 11
- directory names, notation xvi
- DPDEFULT 34, 159
- DPESTRES 34, 159
- DSEDEFULT 34, 159
- DSESTRES 34, 159
- DSICLD 124
- DSIMSG 124
- DSIPARM 124
- DSN3SATH exit 92
- DTLDS 34, 165

## E

- EDI 53, 173
  - action types 177
  - application capture modification
    - samples 55, 62, 67, 72
  - calling 54, 61, 66, 71
  - data types 177
  - data verification 54, 61, 66, 71
  - error handling 56, 62, 68, 73
  - format types 177
  - modifying the source 56, 63, 68, 73
    - examples 56, 63, 68, 73
    - usage notes 57, 63, 69, 74
  - parameters 54, 61, 66, 71
  - starting and ending 54, 61, 66, 71
    - example 55, 61, 67, 71
  - tracing call statements 57, 64, 69, 74
- EDR 35, 161, 167
  - RMF 115
  - select table 141
- EDR event file 31, 163
- EDR select table
  - GROUP statement 142
  - SELECT statement 142
- education
  - See* Tivoli technical training
- EIF
  - event logging 139
  - security 12, 20
  - select table 139
  - starting 140
  - stopping 140
- EIF configuration file, updating 198
- EIF event 133
- EIF event logging 197
- EIF events 137, 196
- EIF probe
  - configuration requirements 12
  - rules 6
- EIF probe rules files
  - description 27
  - include structure 27
  - installing 28
- EIF REFRESH 140
- EIF select table 31, 35, 137, 163, 167, 196
  - DEFAULT statement 137
  - GROUP statement 137
  - ONSELECT statement 137
  - SELECT statement 137
- EIF START 140
- EIF STATUS 140
- EIF STOP 140
- EIF\_LOG 140
- EIF\_LOG\_LEVEL 140
- environment variables, notation xvi
- event
  - generating 145
  - records, format 138
- event distributor 5
  - DD names 195
  - message selection and filtering 137, 196
  - starting 170
  - startup parameters 194
  - stopping 170
  - updating JCL 198
  - updating startup parameters 199
- event feed customize 144
- event feeds 45
  - customizing for NetView for z/OS 128
  - defining additional 143
  - overview 7
- event format 138
- Event Integration Facility (EIF) 196
  - events 137
- event logging, disabling 109
- event processing 6
- Event Pump
  - naming conventions 11
  - program directory 17
- event severity 151
- event-driven registration 133, 141
- Event-Driven Registration 35, 167
- events 196
  - capture of 5
  - DB2 93
    - excluding 94
    - disabling 93
    - enabling 93
    - exceptions 6
    - filtering 5, 137, 196
    - format 137
  - IMS
    - event processing 103, 109
    - event processing, customizing 109
    - excluding 112
    - monitored with a timer 111
    - subsystem up, down, and abend events 110
  - mapping 11
  - messages 6
  - processing 6
  - severities 6

- events (*continued*)
  - sources 11
  - states 6
  - System Automation for z/OS
    - flow 59
    - processing 59
    - scenario 59
  - types 6
  - user-defined 143
- examples
  - application capture modification 55, 62, 67, 72
  - command line interface 152
  - modifying the source with the EDI 56, 63, 68, 73
  - starting and ending EDI 55, 61, 67, 71
  - System Automation for z/OS event processing 59
- External Data Interface 173
- external data interface (EDI) 6
- External Data Interface (EDI) 57
- external data interface, automation 53
  - data verification 54, 61, 66, 71
  - parameters, additional 54, 61, 66, 71
- external data interface, automation, configuring the event feed
  - configuring event feeds 54, 60, 66, 71
- External Integration Facility (EIF), configuring 137, 198

## F

- filtering 35, 167
- filtering events 137, 196
- filters 5
  - Health Checker 101

## G

- GLOBAL statement 133
- GPZ\$EXCL 94, 159
- GPZ#EXCL command 96
- GPZBTBND 92
- GPZBTBND job 92
- GPZCDB2C 87, 159
- GPZCFILE 87, 159
- GPZCGRP 87, 159
- GPZCMAS 87, 159
- GPZCMISC 87
- GPZCPED 34, 159
- GPZCPED 34, 159
- GPZCTAN 87, 159
- GPZDPEDE 34, 159
- GPZDPEST 34, 159
- GPZDSEDE 34, 159
- GPZDSEST 34, 159
- GPZDSPC 31, 36, 160
- GPZMCPSM 34, 160
- GPZMDB2 34, 160
- GPZMHCK 34, 160
- GPZMMCIC 34, 160
- GPZMMDB2 34, 160
- GPZMMZOS 34, 160
- GPZMSMS 34, 159, 160
- GPZPARMS 31, 160
- GPZPUMP 31, 160
- GPZRMFCU 34, 160
- GPZRMFCU sample 119
- GPZRMFDE 34, 160
- GPZRMFST 34, 160
- GPZSHDEX 160

- GPZSHDLR 160
- group entries 191
- GROUP statement 133, 137, 142
- GTMDSPC 160, 163, 168
- GTMEDIAF 53, 173
- GTMEDIAO 65, 173
- GTMEDIOP 70, 173
- GTMEDISA 173
- GTMEDITB 54, 61, 66, 71, 160, 173
- GTMEIFSL 35, 167
- GTMMON statement 34
- GTMPUMP 160, 163, 192, 193, 194
- GTMSAM02 160
- GTMSAM03 160
- GTMSAM07 31, 160, 163
- GTMSAM08 160, 163
- GTMSAM09 160
- GTMSAM10 31, 160, 163
- GTMSAM12 160
- GTMSAM13 163
- GTMSAM22 31, 115, 160, 163
- GTMSAM23 31, 35, 161, 163, 167
- GTMSAM24 31, 36, 161, 163, 168
- GTMSAM25 31, 141, 161, 163
- GTMSRVR 160, 163, 195

## H

- handler 146
- Health Checker 101
  - filters 101
  - monitor file 34
- HZSFMSG service 101

## I

- IBM Tivoli AF/OPERATOR 53
- ICICPARM 34, 149, 160, 161
- IDB2PARM 34, 149, 160, 161
- IEFACTRT SMF exit 163
- IEFC452I processing 49
- IHCKPARM 34, 102
- IHS\$CAT2 161
- IHS\$CMD 161
- IHS\$CMD2 161
- IHS\$DISC 161
- IHS\$EXCL 161, 162
- IHS\$MAT 161
- IHS\$MPF 161
- IHS\$MPF exit 105
- IHS\$OPF 161
- IHS\$PRM 109, 126, 161, 162
- IHS\$PRMU 109
- IHS\$RDIS command 96
- IHS\$UPRM 162
- IHSIAOJ 161
- IHSIAOJ job 106
- IHSIAOE0 106
- IHSIAOEC 106
- IHSIAOEE 161
- IHSIAOEE table 106
- IHSIAOEJ 161
- IHSIAOML job 106
- IHSIAUTn 12
- IHSIMDBS 112
- IHSIMMSC 112

- IHSIMPGM 112
- IHSIMTXN 112
- IHSIMTXQ 112
- IHSIPRM 162
- IHSISUB6 112
- IHSSMONI 60, 161
- IHSSOBSX exit 59
- IHSSSTRT program 60
- IMS
  - AO controller exit 161
  - AO exit 106
  - AO exits 185
    - IHSIAOE0 185, 186
    - IHSIAOEC 185, 186
    - interface 185
  - event logging, disabling 109
  - event processing
    - customizing 109
    - description 109
    - subsystem up, down, and abend events 110
  - excluding resources and events
    - exclude rules 113
    - process 113
    - reasons 112
  - installing 123
  - installing components 104
  - integration 103
  - modifying setup 104
  - OTMA, enabling 104
  - polling processing 103
    - customizing operation and thresholds 111
    - description 110
    - limiting events generated by polling monitors 111
    - polling timers 112
    - viewing polling monitor tasks 112
  - processing overview 103
  - security 13
  - security, configuring 105
  - subsystem monitoring 103
  - tuning 113
  - verification 108
  - z/OS components, configuring 19
- IMS subsystem data resources 103
- IMS\$MPF member 19
- include file structure 27
- installation
  - results 18
  - task definition 17
- installing
  - EIF probe 23, 24
  - EIF probe rules files 28
  - Netcool/OMNIBus 23
  - rules files 28
  - schema 25
  - System Automation for z/OS 58
- integration
  - IMS 103
- ISHIMMTO 112
- ISMSPARM 34, 121, 149, 159, 160

## J

- JCL IEF452I error message processing 49
- JCL symbolics in naming conventions 11
- JES JOBLOG 47
- JES2 49
- JES3 49

- JOB statement 133
- JOBLOG 47

## L

- language support, setting parameters 195
- log data sets
  - allocating 21
- logging, EIF event 139, 197
- lookup tables 151
  - adding rows 152
  - deleting rows 152
  - displaying rows 151
  - updating rows 152
- LU6.2, common parameters 194

## M

- Manager, CICSplex System 85
- manuals
  - See* publications
- master address space 4
  - starting 40
  - stopping 40
- MCICPARM 34, 160
- MDB2PARM 34, 149, 160
- message mask member 149
- message selection 35, 167
- messages
  - event severity 151
  - IMS 103
- migration
  - object pump 189
  - registration data 190
  - scenario 191
  - stand-alone operation 189
  - TBSM Version 3 object pump 189
- modifying
  - triggers 26
- monitor file 34
- MONITOR statement 133
- monitoring
  - DB2 subsystem 91
  - IMS subsystem 103
  - resources
    - storage management 121
- monitoring level
  - System Automation for z/OS 60
- MPFLSTxx member 19
- MVS
  - MPFLSTxx 19
  - preparation 19
  - PROGxx 19
- MZOSPARM 34, 149, 160

## N

- name\_value\_pairs 138, 196
- NAT IP 195
- Netcool/OMNIBus
  - Administrator, starting 151
  - alerts 6
  - component administration 151
  - EIF probe configuration 29
  - EIF probe installation 23, 24
  - EIF probe overview 6

- Netcool/OMNIBus (*continued*)
  - EIF probe rules files 27
  - installation overview 23
  - installing EIF probe rules files 28
  - object server configuration 26
  - ObjectServer server overview 6
  - probe configuration 29
  - rule file include structure 27
  - schema installation 25
  - schema updates 25
- NetView
  - command authorization table 161
  - command definitions 161
  - MAT entries 161
  - operator definitions 161
- NetView for z/OS 57
  - configuring 123
  - customizing event feeds 128
  - installation overview 123
  - Kanji messages and help 124
  - parameter members 124
  - security 126
  - started tasks 123
  - tuning 127
  - using event feeds 123
- NetView start procedure 58
- Network Address Table (NAT) 195
- notation
  - environment variables xvi
  - path names xvi
  - typeface xvi

## O

- object pump
  - migration to Event Pump 189
- OMNIBus
  - See* Netcool/OMNIBus
- OMVS segment 20
- ONSELECT statement 137
- operation, stand-alone
  - migration 189
- OPS/MVS EDI interface
  - installation 69
- OTMA, enabling 104
- overview 3

## P

- parameters
  - data space startup 36, 168, 193
  - NetView for z/OS 124
- PARMDSPC 31, 160, 163
- PARMDSPC startup parameter 36, 168
- PARMEIF 31, 36, 161, 163, 168
- PARMPUMP 160, 163
- PARMSVR 163
- path names, notation xvi
- polling
  - events generated by polling monitors 111
  - IMS, refreshing exclude rules 113
  - timers in IMS 112
  - viewing monitor tasks 112
- preinstallation tasks
  - applying required PTFs and APARs 17
  - getting the PSP Bucket 17

- preinstallation tasks (*continued*)
  - program directory 17
- preventive service planning 17
- PRMEDRSL 31, 141, 161, 163
- PRMEIFSL 31, 35, 137, 161, 163, 167
- probe rules 6
- probe rules files
  - description 27
  - include structure 27
  - installing 28
- program directory 17
- program-to-program interface (PPI)
  - System Automation for z/OS 58
- PROGxx member 19
- PSP 17
- PTF 17
- publications
  - accessing online xiv
  - Event Pump for z/OS xiii
  - ordering xiv
  - related xiii

## R

- REFRESH, EIF 140
- REGDATA 31, 115, 160, 163
  - overview 133
- registration 133
- registration data 31, 133, 163
  - activating new data 191
  - adding group entries 191
  - converting captured data 190
  - enabling data capture 190
  - JOBLOG data source 49
  - migration 190
  - migration scenario 191
- registration data file 115
- resources
  - based on events 11
  - DB2, excluding 94
  - detecting 11
  - IMS, excluding 112
  - storage management 121
  - supported 94
  - system-managed storage 121
  - types 11
- REXX alternate runtime library 32, 164
- RMF
  - EDR select table statements 115
  - exceptions 34
  - GPZPARMS parameters 115
  - monitor file 34
- RMF statement 133
- RMF\_SERVER\_INTERVAL 156
- RMF\_SERVER\_IP 156
- RMF\_SERVER\_PORT 156
- RMF\_SERVER\_TIMEOUT 156
- RMFDEFLT 34, 160
- RMFDEFLT file 115
- RMFSTRES 34, 160
- RMFSTRES file 115
- rules files
  - description 27
  - include structure 27
  - installing 28

## S

- samples 159
- SCEERUN 19
- schema installation 25
- schema updates 25
- security
  - NetView for z/OS 126
- security requirements 12
- SELECT statement 137, 142
- select table, EIF 137, 139, 196
- SETPROG command 20
- SGTMMODS 19
- SGTMSAMP data set 32, 164, 166
- SGTMSAMP library 31, 163
- slots 138
- SMF exit, IEFACTRT 163
- SMF ID 12
- SMF record type 30 automation 163
- SMS
  - exceptions 34
  - message mask 149
  - messages 34
  - monitor file 34
- SMS resources
  - storage reference 121
- SMS statement 133
- SOAP requests 12, 20
- SOAP server
  - SMF ID 12
- source collector 5
  - starting 170
  - startup parameters
    - dual mode 192
  - stopping 170
- SOURCE statement 134
- SQL commands 151
  - command line interface 152
- stand-alone operation
  - migration 189
- START, EIF 140
- starting
  - data space 39, 169
  - event distributor 170
  - master address space 40
  - Netcool/OMNIbus Administrator 151
  - source collector 170
  - three address spaces 169
  - two address spaces 39
- startup parameters
  - data space 36, 168, 193
  - event distributor 194
  - source collector for dual mode 192
  - TBSM Version 3 and Event Pump Version 4 192
- startup procedures
  - three address spaces 163
  - two address spaces 31
- state lookup table
  - See lookup tables
- STATUS, EIF 140
- STC messages 49
- STEPLIB 124
- STOP, EIF 140
- stopping
  - data space 39, 169
  - event distributor 170
  - master address space 40
  - source collector 170

- stopping (*continued*)
  - three address spaces 169
  - two address spaces 39
- storage management
  - monitoring 121
- subsystem monitoring
  - DB2 91
  - IMS 103
- symbolic naming conventions, JCL 11
- syntax conventions xvi
- SYSLOG 50
- sysplex
  - EDR traps 141
  - SOAP requests 12, 20
  - TCP/IP considerations 12
- System Automation for z/O
  - installation overview 58
- System Automation for z/OS
  - event flow 59
  - event processing 59
  - event scenario 59
  - initialization 60
  - monitoring level 60
- System Manager, CICSplex 85
- System Modification Program Extended (SMP/E) 91, 104, 123

## T

- table, EIF select 137, 196
- TBSM Version 3 object pump
  - migration to Event Pump 189
- TCP/IP libraries 32, 164
- TCP/IP, common parameters 194
- TCPIP
  - security 12
- TCPIP services 20
- TEC\_generic 137
- terminology xiv
- three address spaces
  - configuring 163
  - starting 169
  - stopping 169
- thresholds
  - IMS 103
    - customizing 111
- Tivoli Information Center xiv
- Tivoli technical training xv
- Tivoli user groups xv
- trace
  - EDI call statements 57, 64, 69, 74
- training, Tivoli technical xv
- triggers, modifying 26
- tuning
  - NetView for z/OS 127
- two address spaces
  - configuring 31
  - starting 39
  - stopping 39
- typeface conventions xv
- types, resources 11

## U

- user defined slots 196
- user groups, Tivoli xv
- user-defined events 143

## **V**

variables, environment, notation xvi  
verifying  
    installation 39

## **Z**

z/OS components 4  
z/OS event record 137







Product Number: 5698-B34

Printed in USA

SC27-4042-00

